1997

A testing advisor and optimization pre-constrainer to reduce nitrogen oxide emissions from wall fired boilers

Steven W. Steele
Lehigh University

Steel, Steven W.

A Testing Advisor and Optimization Pre-Constrainer to Reduce Nitrogen Oxide Emissions from Wall Fired

June 1, 1997

A Testing Advisor and Optimization Pre-Constrainer to Reduce Nitrogen Oxide Emissions from Wall Fired Boilers

by

Steven W. Steele

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in candidacy for the degree of

Master of Science

in

Mechanical Engineering

Lehigh University

May 7, 1997

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Mechanical Engineering.

4/28/97

Date

Thesis Advisor

Chairperson of Department

Acknowledgments

I would like to thank Dr. E. Levy for giving me the opportunity to study under him at the Energy Research Center. I would also like to thank my wife for her support and motivation. I would also like to thank my Mother and Father for their help and support while completing my education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Since the 1960's, it has been known that energy production from fossil fuels results in the release of pollutants which are harmful to the surrounding environment and can cause health problems in people. These pollutants include nitrogen oxides, sulfur dioxides and many others. As a result, in 1990, amendments were made to the Clean Air Act, which requires power plants to take several steps to reduce nitrogen oxide emissions as well as other pollutants.

Previous studies on NOx control have shown that nitrogen oxide emissions from utility boilers can be reduced by operating the boiler at certain optimum control settings. These optimum control settings can be determined by conducting parametric tests on the boiler and analyzing the data.

An optimization testing advisor, using an expert system and knowledge based on testing previously performed by the Energy Research Center, has been developed to help simplify this testing procedure and help electric utilities reduce NOx emissions. The purpose of the advisor is to guide a plant operator or engineer, who may not be familiar with performing boiler optimization, through the process of conducting the parametric tests needed to optimize boiler operation for reducing NOx and improving unit heat rate.

The testing advisor was developed for general use with nearly any wall-fired boiler. The knowledge it is based on follows the general trends and methods for reducing NOx. It is capable of testing a boiler with any number of mills and can be used on a boiler either with or without overfire air. It can also handle boilers with single or dual register burners and multiple rows of overfire air ports.

A second software tool called the Optimization Pre-Constrainer was also developed to help the optimization routines find better and more consistent results each time optimal control settings are to be determined. The pre-constrainer is a DOS based program coded in C and is based on testing performed at several power plants. The pre-constrainer was also developed for general use; although it is currently incorporated in advisors for tangentially fired boilers with both conventional and low NOx burners.

INTRODUCTION

As early as the 1960's, there has been a growing concern over nitrogen oxide emissions. Since that time, evidence has been gathered to show that NOx causes photo-chemical smog and acid rain, both of which are harmful to the environment and possibly to our health. In response to these potential dangers, the Federal Government made changes to the Clean Air Act in 1990. These changes require power plants to meet certain emissions standards or else pay substantial fines [11]. This thesis discusses two software tools developed to help power plants reduce NOx emissions.

NOx, which includes all nitrogen oxide radicals, is created from the combustion of fossil fuels such as coal. Nitrogen oxide (NO) constitutes 95 percent of all NOx while NO_2 constitutes the rest [3]. NOx is generally divided into two categories. The first category is **thermal NOx,** which refers to the NOx created from the nitrogen in the combustion air. The second is **fuel NOx**, which refers to the NOx created from the nitrogen in the fuel [1].

Since NOx control has become a major issue, many theoretical models based on chemical kinetics and experimental studies have been performed to determine how to best reduce NOx emissions. There are currently two types of NOx control techniques. The first is post combustion NOx control which includes selective non catalytic reduction (SNCR), catalytic decomposition and other methods that try to remove NOx from the flue gases after it has already been formed [2]. The second includes techniques to reduce NOx during the combustion process such as low NOx burners and overfire air.

Working with the Potomac Electric Power Company and the Electric Power Research Institute, the Energy Research Center (ERC) developed procedures for reducing NOx emissions through boiler optimization [12, 21, 22, 23]. This technique of reducing NOx has not only been shown to be very effective but has other advantages such as large savings from not having to spend money on boiler modifications and better boiler efficiencies.

The ERC has determined the parameters which affect NOx formation, safety limits and boiler efficiencies [21,23]. Using this information, quantitative test data, and heuristic knowledge from several research engineers at the ERC, an expert system incorporating this knowledge was developed to help utilities reduce NOx emissions.

This thesis discusses in detail the development of two separate software packages which help reduce NOx emissions. First, an expert system software package for testing wall-fired pulverized coal boilers was developed. The expert system acts as an experienced test engineer that guides the plant operator or engineer through the testing strategy to optimize a boiler. The second, the optimization pre-constrainer is a DOS based program intended to be used in conjunction with a neural network/optimization code developed at the ERC to determine optimal boiler control settings.

The knowledge base of the expert system contains qualitative data regarding wall-fired boilers and the techniques used to reduce NOx emissions from these boilers. This knowledge was acquired through the experience of the personnel at the ERC.

An expert system is a computer problem solving tool which is capable of emulating human expertise in a narrow domain [8]. Expert systems are different from conventional

computer programming languages in that they generally operate on qualitative data, whereas a conventional language is best with numbers. Expert systems are also different in that they are based on heuristics or rules of thumb. Conventional programming languages are very structured and always perform the same way every time [9].

The wall-fired expert system is interactive and asks the user to supply a large amount of both qualitative as well as quantitative data. It is also interactive when it recommends a test or when a violation or warning occurs. Currently the wall-fired expert system is only capable of testing at full load conditions, but it should require only small changes to be adapted for low load conditions. It is also generalized, so it is capable of testing nearly any wall-fired boiler.

The development of the expert system is described in detail in this thesis. This includes the development of the knowledge base and all of the corresponding modules, the overall operation of the system, and the operation of each individual module. Finally, simulated results and recommendations for future enhancements are discussed.

The logic for the optimization pre-constrainer is based on analysis of test data obtained from several boilers. The pre-constrainer deals primarily with quantitative data and performs a definite set of calculations each time it is run, so it was developed in C, a conventional programming language. The pre-constrainer is non-interactive and it is intended to be transparent to the user. The development of the optimization pre-constrainer, including the logic behind its method, it operation and its verification are described.

EXPERT SYSTEMS

While humans have dreamed about creating an intelligent machine or robot, it was not until this century that these dreams have become possible due to the development of the computer, possibly the greatest invention of this century.

Computers have been around since World War II, performing routine numerical calculations and storing or manipulating large amounts of data. Although such activities are useful, it cannot be said that these activities are in any way intelligent or involve intelligence. Intelligence is the ability to learn from experience; the ability to acquire new knowledge and use this knowledge to solve problems [5]. Routine computations are clearly not intelligent since they simply follow a definite algorithm to arrive at the answers the same way every time.

In 1950, Alan Turing published his paper, "Can a Machine Think," which led to many people speculating as to whether computers could be used to solve problems which humans could not [5, 7]. This speculation led to many experiments to see if machines were intelligent. One such experiment, proposed by Turing, suggested that if a human talks to a machine and does not realize he is not talking to another human then the machine could be considered intelligent [7]. Thus, if a machine can emulate human intelligence, it can be considered intelligent.

Within a few years, computer programs to play chess, checkers, and to solve mathematical proofs, which were previously only done by man, were now being done with computers. This eventually led to the field we know today as artificial intelligence (AI).

As the field of AI was just beginning, research was quickly classified into two distinct categories. The first category is neural networks, and the second is knowledge based systems. Neural networks resulted from years of studies to determine the biological basis for human intelligence [5]. Neural networks are based on a model of the brain with receptors, neurons and the nervous system [5]. Artificial neural networks are currently being used in applications such as image processing, financial predicting and voice recognition [6, 22].

The second category includes knowledge based systems or expert systems. Knowledge based systems resulted from studies of human intelligence which focused on a behavioral viewpoint instead of a biological one [5]. This area focused on humans' ability to reason, learn, and solve problems. Obviously, for machines to be useful to us, they must be able to solve difficult problems, which humans cannot solve. For these reasons, the problem solving methods used by humans became a much researched topic.

We now know that humans employ several methods of solving problems. The simplest and least interesting method is called the algorithmic method. This method is simply following a set pattern or procedure to solve a problem the same way every time. However, many problems do not have such simple algorithmic solutions, and other methods must be employed to arrive at a solution.

A second method is to generate all possible solutions of a problem, then throw out any solution which violates a rule or constraint. Many early expert systems used this approach to solve problems, including chess and checkers [5]. In both cases, the computer determined all the possible moves and then chose the next move by the one

which was found to produce the best scenario. Although this approach was somewhat successful, these games could still be beaten by human players.

The third and most complicated method includes techniques which rely on some sort of knowledge to generate only possible valid and reasonable solutions. This approach does not generate every combination of possible solutions, but only those which are most logical. This method led to what we now call knowledge bases or expert systems.

Expert systems use rules collected from experience or other sources to determine a solution to a problem. An example is an auto mechanic. Auto mechanics do not test every component of the engine to determine what is wrong, but instead quickly analyze the situation and only test components that are most likely to be the problem. These components are based on the mechanics' knowledge about engines and past experiences with engine repair.

Knowledge based systems use general methods and knowledge special to a narrow domain to solve problems. These systems generally have two types of information about a domain, knowledge and heuristics. Knowledge includes facts which are definitely known to be true. Heuristics are rules based on experience, sometimes called rules of thumb. In expert systems, heuristics are aquired from another expert such as a human expert. Heuristics generally lead to a solution quickly and efficiently, but may not always lead to the best or optimum solution. This is considered acceptable in most cases because the optimum solution is usually difficult and time consuming to find.

Expert systems have three main components: the knowledge base, the inference engine and the working memory. The knowledge base is the area of the system which

contains the knowledge and heuristics pertaining to the domain. The knowledge base is created by the creator of the expert system. The knowledge in the knowledge base can be represented in many different ways. It can be represented in a database fashion, frame based or object oriented fashion, or rule based. One of the most popular methods to represent the knowledge is by rules. Rules are statements which read as follows, if *premise* then *action*. Rules are an excellent way to represent heuristics and are very easy to understand and modify. An example of a rule is, "If LOI high then increase O_2." If a fact exists in the system stating LOI is high, then this rule will fire and some action will be performed. This method of searching the facts in the system and comparing them to the premise clause is called "pattern matching" [7]. A fact is a true statement about something. An example of a fact is "LOI is HIGH." When a fact matches with a rule, the rule is fired. When a rule is fired it means the rule is executed and some action is performed which usually modifies the state of the system.

Another representation scheme which has grown in popularity is the object orientated approach. This method represents objects in a way very similar to how humans describe objects. For example, if an object called "car" is described, it generally has several attributes to describe the car. Such attributes include color, size, weight, horse power and others. Benefits of object orientated systems include their ease of development and modification [7].

The working memory is the second portion of an expert system. It is simply the program's memory in the computer used to store the current state of the system. The current state includes the facts and the rules which are on the agenda to be fired. Once a

rule is fired, the working memory is modified with other rules to be fired or with new facts.

The inference engine is the third component of an expert system. The inference engine uses the rules and facts in the working memory to determine which rules are to be fired to reach a solution. The inference engine is basically the method or reasoning process used to solve a problem. The figure shown below shows a diagram of a typical expert system.

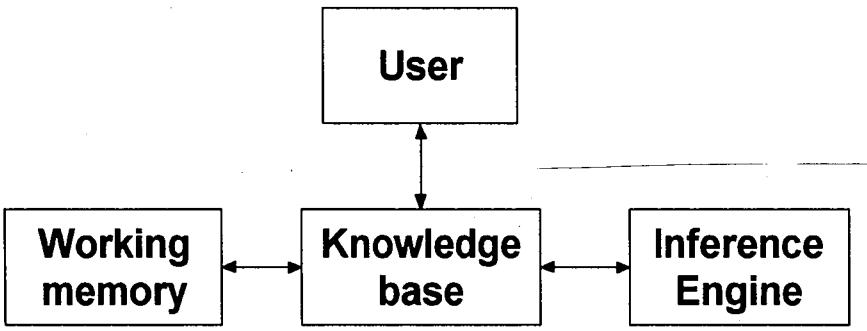

Figure 1 - Components of an expert system

The inference engine can use two different methods to arrive at a solution, these methods are forward chaining and backward chaining. Forward chaining is a method which collects all the necessary information about a domain at the start and then uses this information to find a solution. Thus, the solution is not known at the start. Backward chaining starts with an assumed solution at the start and then works backward to determine if the information available does produce the assumed solution.

Today, thousands of expert systems have been developed and are used successfully in solving complicated problems. One of the earliest and most successful expert systems is

DENDRAL, a forward chaining rule based expert system used to identify the molecular structures of molecular compounds [6,9]. This system eliminates millions of possible solutions by throwing out compounds which could not possibly be a compatible compound. DENDRAL is based on heuristics which chemists use to do the same thing. DENDRAL is so successful that it has been said to do a better job than humans in some cases. Another expert system, and probably the most well known, is MYCIN, a backward chaining rule based system used to help doctors diagnose infectious blood diseases [6,9].

Expert systems in common use today are found in applications such as speech recognition, grammar checking, world wide web search engines and in many engineering applications [8,9].

Several expert systems are either in development or have been developed for use in the electric power industry. Some examples include CRAFT, which is a power system restoration advisor, and SARA, a nuclear safety review advisor. There are many others which have been developed and are in use. A few expert systems to help power plants reduce or control NOx emissions have been developed and are in use today. NOxSMART, an on-line expert system advisor, monitors on-line plant data and recommends control settings to reduce NOx emissions [8]. Working with the Potomac Electric Power Company and the Electric Power Research Institute, the ERC recently developed an off-line expert system testing advisor for use on conventional boilers and low NOx tangentially fired boilers. Coal quality expert (CQE), a project funded by the Department of Energy and EPRI, is an expert system to determine the most cost effective coal for acceptable plant performance and emissions [12].

Expert systems, like conventional programming languages, are developed using specific tools. Conventional programming languages use editors and compilers, while expert system are usually created using expert system shells. Expert system shells are essentially completed expert systems with the knowledge base missing or removed. The creator of the expert system must develop the knowledge base and place it into the shell. Once the knowledge base is incorporated, the shell becomes a new expert system in itself.

The wall-fired testing advisor in this thesis was developed using CLIPS v6.0. CLIPS is an expert system shell developed by the Software Technology Branch at NASA and uses the forward chaining method in its inference engine. CLIPS is a combination of a rule based system and an object orientated system [10].

CLIPS was chosen as the development tool for this project for several reasons. First, a large portion of the knowledge base will be heuristic in nature, and many facts about the system are qualitative. Rule based systems are very efficient and easy to maintain with heuristic knowledge and qualitative data. Second, a large number of the data pertaining to a boiler is quantitative. Object oriented systems are very efficient with quantitative data and can be combined with heuristics, so pattern matching can be performed on objects. CLIPS was also chosen because the testing advisor gathers a large number of data initially and then guides the user through the testing procedure. This causes the forward chaining method to be the most appropriate in this case. Another advantage of CLIPS is its ability to divide the knowledge base into modules or sections. This is important because it allows a large knowledge base to be broken down into smaller, more manageable parts. Finally, CLIPS is tightly integrated with the C

programming language, which allows the expert system to call external applications written in C. This integration is desirable so a user interface and external data analysis programs could be used by the expert system.

The knowledge base of the wall-fired testing advisor and its software description is described in the following sections.

WALL-FIRED BOILER TESTING ADVISOR

Introduction

The purpose of the testing advisor is to guide the user through a series of tests which will be used to determine the optimum operating conditions of the boiler at various NOx levels. The user of the testing advisor is expected to be a plant operator or an engineer knowledgeable in boiler operations, but not particularly knowledgeable in NOx reduction techniques and boiler testing. The user is also expected to be knowledgeable enough to make qualitative judgments about the boiler, such as flame quality and water wall cleanliness. Thus, the testing advisor is designed to guide the user step by step through a series of tests to optimize the boiler, while maintaining good boiler performance and safe operating conditions.

The advisor was designed to perform two different optimizations. The first is to determine the conditions needed to achieve the lowest NOx emissions. The second is to determine the conditions which minimize heat rate at a target NOx. Past ERC studies have shown that large reductions in NOx are possible through boiler optimization. During several weeks of testing at Chalk Point Unit 2, baseline NOx emissions were reduced from 0.9 lb/MBtu to 0.63 lb/MBtu at full load, 355 MW [21].

Parametric testing of the control variables known to affect NOx formation is used to gather the database required for optimization. These control variables can be identified through theoretical studies of NOx formation, or by directly testing the boiler.

There are two classifications of NOx formation, thermal NOx and fuel NOx.

Thermal NOx refers to the NOx formed from the nitrogen in the combustion air. Fuel NOx is the NOx formed from the nitrogen bound in the fuel. Many theoretical models have been developed to study the variables which control NOx formation. These models consider both the thermodynamics and kinetics of the three principal NOx formation equations.

The three principal reactions occurring during the formation of thermal NOx are:

$$N_2 + O \rightarrow NO + N \quad (1)$$

$$N + O_2 \rightarrow NO + O \quad (2)$$

$$NO + O \rightarrow NO_2 \quad (3)$$

These models have shown that as the flame temperature rises, the formation of NO increases, while the NO_2 decomposes to form more NO. Molecular nitrogen (N_2) and oxygen (O_2) in the combustion air dissociate as the temperature rises, thus forming more NO [1, 2, 17]. Thus, the total concentration of NOx increases as the flame temperature rises.

Analysis of the chemical kinetics of the three principal equations has shown that the rate of NO formation is also a strong function of reaction time [3]. Other than the flame temperature and the reaction time, the stoichiometric ratio of fuel is also an important variable in the formation of thermal NOx.

Fuel NOx is the major source of NOx emissions from nitrogen rich fuels, such as coal and oil. Studies have shown that fuel NOx contributes approximately 80 percent of the total uncontrolled emissions when burning coal [3]. Experiments have also shown that

the reaction rate is much faster for fuel NOx than it is for thermal NOx [16]. Experiments have also shown that the local fuel/air stoichiometric ratio is important in the formation of fuel NOx [3].

Using results from both theoretical and experimental analysis, several methods or rules for NOx control have been developed. Since NOx formation tends to increase as the flame temperature increases, decreasing the flame temperature will decrease NOx. NOx formation needs excess oxygen so that the stoichiometric ratio of air/fuel is greater than one. Thus, decreasing the excess oxygen will reduce NOx formation. It has also been shown that having a fuel rich flame and later introducing more air to complete combustion decreases NOx formation. This technique is called staged combustion and is the driving theory behind the use of overfire air for NOx control [2]. Lastly, the fuel with the least nitrogen in it should reduce NOx emissions.

Many experiments on different types of pulverized coal fired boilers have shown that when the methods above are implemented, NOx emissions are reduced. These tests have shown that staged combustion, decreasing the flame temperature, limiting the oxygen for combustion, and fuel quality are important factors in NOx formation [21,23].

Using these general rules on NOx formation in boilers, each boiler operating parameter must be tested in a systematic way. The optimization process relies on a series of parametric tests conducted on each boiler control parameter.

Although this advisor was developed based on the configuration and test data for Chalk Point Unit 2, it has been designed to be flexible and general for a wide variety of wall fired boilers. At Chalk Point Unit 2, several parameters including economizer oxygen

level, overfire air flow, secondary air shroud opening and register settings, and mill coal flows have been determined to affect NOx formation [21].

In addition to the control parameters which affect NOx formation, other parameters affecting boiler performance and safety are also considered and monitored during the testing process. During testing, safety is of the utmost importance. All parameters affecting safety such as furnace oxygen level, windbox pressure, CO level and steam temperatures are closely monitored to make certain the boiler is operating safely.

During testing, performance parameters are also monitored to assure efficient operating conditions. One example is LOI, or loss on ignition. This parameter, which indicates the amount of unburned carbon in the fly ash, is an indicator of boiler performance.

The advisor, like other boiler advisors developed at the ERC, was developed using CLIPS 6.0. The advisor is an expert system which is based on theoretical and experimental results on the factors which affect the operation of a wall fired boiler. The advisor relies on a series of rules for conducting each test for each parameter in a safe and efficient manner. Once the advisor has guided the user through the series of tests, the data are processed by an external neural network-optimization program, and analyzed to determine the optimal operating conditions for the boiler being tested.

Each parameter discussed above, safety and performance issues, the logic used and a description of the software will be discussed in the next two sections.

Knowledge Base

Based on data gathered at Chalk Point Unit 2 and the models discussed in the Introduction, several parameters have been determined to control NOx formation. These parameters are economizer oxygen level, overfire air register opening, mill coal flow distribution, secondary air shroud opening, and secondary air register settings.

The economizer is a counterflow heat exchanger, located downstream from the furnace, which recovers heat from the flue gas and uses it to heat the feedwater into the boiler [4]. The percent excess oxygen in the flue gas in the economizer is measured using oxygen sensors. Although this is not the true furnace oxygen, it is a good estimate considering it is difficult to get a true reading in the furnace itself.

Chalk Point Unit 2 has an opposed wall-fired double reheat boiler with three rows of Riley Stoker Low NOx burners on both the front and rear walls. Each elevation has four burners on each wall, fed by Babcock & Wilcox ball and race pressurized mills. Thus, there is a total of six pulverizers. The capacity of each mill is measured in terms of thousands of pounds of coal per an hour (KLb / HR). The pulverized coal is carried from the pulverizers to the burners in the primary air flow.

To achieve further NOx reduction, this boiler is also equipped with overfire air ports or NOx ports. Both the front and rear walls of this boiler have six overfire air ports. There is one port located eight feet directly above each top row burner and there are two wing ports located between the side walls and the outside burners (see Figure 2). Thus, there is a total of twelve overfire air ports on this boiler. Each overfire air port is

rectangular in shape and is divided into 1/3 and 2/3 sections. The 1/3 and 2/3 sections each have dampers which can be adjusted manually or automatically to control the amount of air in each section. These overfire air ports are not used at boiler loads below 210 MW [19].

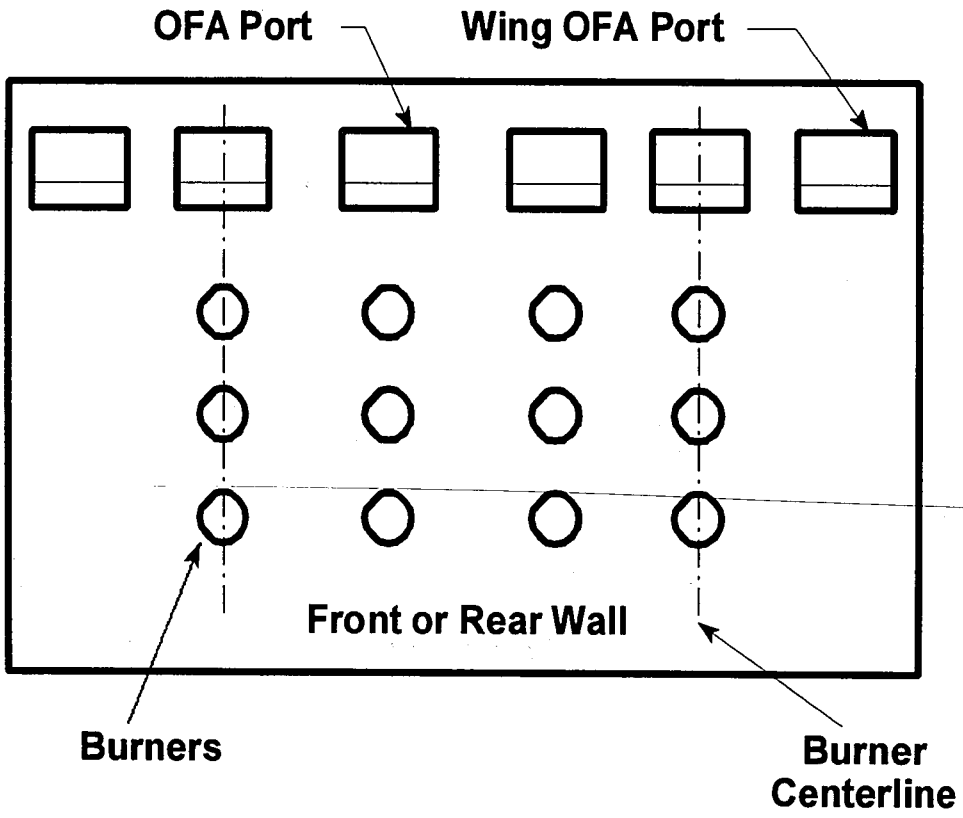

Figure 2 - Diagram of Chalk Point burner wall

In the Riley design, the burner secondary air shroud controls the total amount of secondary air flowing through the burner. Each burner has a shroud which can be either manually or automatically controlled. The shrouds can be open from 0 % (closed) to 100 % (open). When operating at full load conditions, the shrouds are generally maintained

100% open, but can be closed more. Figure 3 below, shows a diagram of the Riley Stoker Low NOx burner.

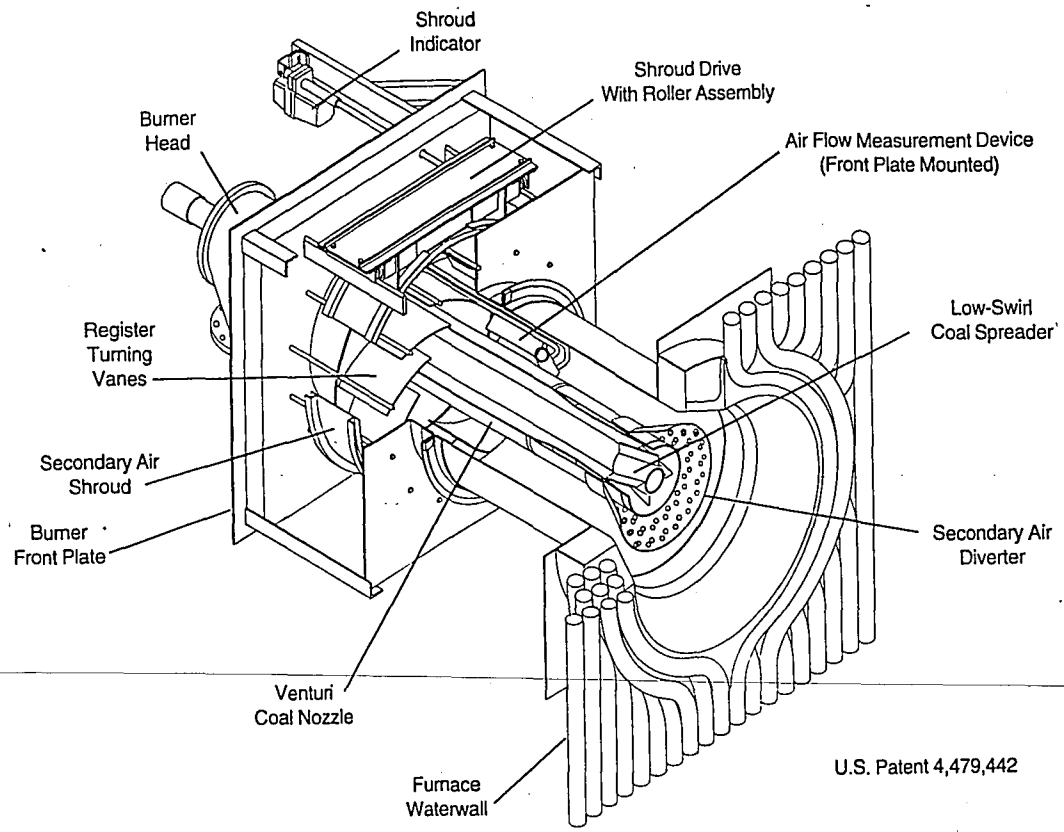

Figure 3 - Riley Stoker Low NOx Burner [21]

The secondary air register swirl vanes control the amount of swirl imparted to the secondary air. Swirl is used to enhance the mixing of coal and air. Each burner has swirl vanes located in the secondary air path which are manually adjusted. The swirl vanes are generally adjusted to improve the flame quality, ignition point of the flame and air velocities. They also have an effect on NOx formation.

Finally, each burner is equipped with a coal spreader which can be adjusted axially in and out. The coal spreader distributes the coal and air mixture out into a wider area.

This promotes mixing of the fuel/air mixture from the coal nozzle with the secondary air. The total air needed for complete combustion consists of the secondary air and the primary air, which carries the coal to the burners.

There are other burner features which Chalk Point does not have, but which are provided by other manufacturers of wall fired burners. Figure 4 below shows a diagram of the Babcock & Wilcox DRB-XCL™ Low NOx burner, which has dual register swirl vanes. In this type of burner, the secondary air path is divided into two sections and each section has swirl vanes. The registers in the outer secondary air path are used to control the air velocities and flame length. The registers in the inner secondary air path are used to adjust the ignition point, flame stability and air balance. Although Chalk Point does not have dual register swirl vanes, the logic needed to handle boilers with these is included in

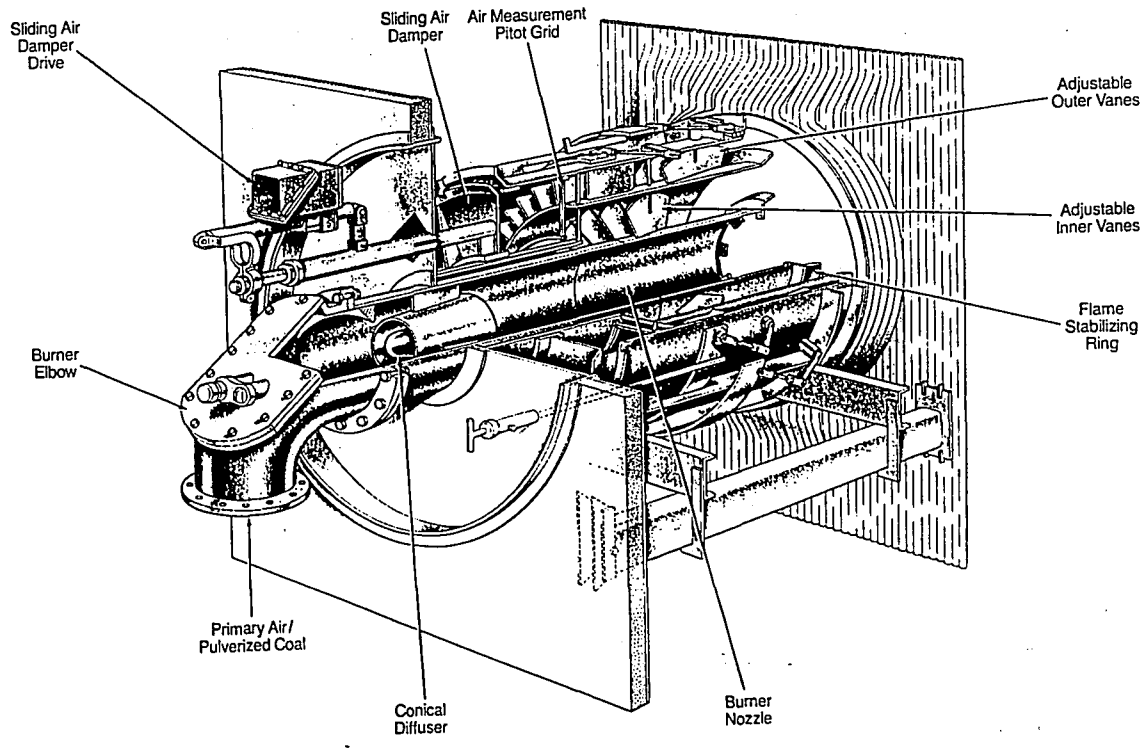

Figure 4 - Babcock & Wilcox DRB-XCL Low NOx Burner [1]

the advisor. This was done to make the advisor as general as possible.

It should be noted that other types of wall-fired boilers also have different types of overfire air systems, or some may not have overfire air at all. The advisor can handle the case where no overfire air exists, and the case where more than one row of overfire air ports exist.

Adjusting one or more of these controllable parameters creates different mixing patterns and combustion conditions. This affects emissions, performance measures and safety. The safety and performance issues are very important and the advisor considers these throughout the testing procedure.

Economizer Oxygen

Whenever complete combustion is desired, it is generally required that excess oxygen beyond the stoichiometric amount is needed. It was shown earlier that NOx is a strong function of economizer oxygen concentration. Test data from Chalk Point also show the same trend that as economizer oxygen decreases, NOx also decreases linearly (see Figure 5). This figure shows that when decreasing the economizer oxygen level from 4% to 3%, the NOx decreased from 0.85 lb/Mbtu to 0.79 lb/Mbtu.

It is desirable to reduce economizer oxygen level as low as possible, but there are disadvantages if it is reduced too far. As economizer oxygen is decreased, LOI, loss on ignition, increases (see Figure 6). Since boiler efficiency decreases as LOI increases, it is usually desired to maintain low LOI levels. Figure 7 also shows that if economizer oxygen is reduced too low, the flue gas CO concentration increases rapidly. Thus, when reducing

economizer oxygen to reduce NOx, the O2 level must not be reduced to levels which cause excessively high levels of LOI or CO.

Overfire Air

Some low NOx firing systems utilize overfire air to reduce NOx emissions. In this case, the total secondary air is divided between the burners and the overfire air ports, creating staged combustion. Staged combustion is a key player in NOx formation. The amount of air to the overfire air registers and to each burner can be adjusted to further reduce NOx. At Chalk Point, opening the overfire air dampers from 0% to fully open (100%) decreased the NOx from 0.989 lb/MBtu to 0.828 lb/MBtu. This clearly shows how important overfire air settings are in affecting NOx (see Figure 8). The distribution of air between the overfire air registers and the burners depends on the system pressure, which is related to the economizer oxygen level. For this reason, the advisor tests different overfire air settings at several different oxygen levels. Although use of overfire air is beneficial, it also has disadvantages. Use of overfire air can reduce the steam temperatures, which affect performance. Opening the overfire air dampers too much also causes LOI to increase.

Secondary Air Shroud

The total amount of secondary air to the burners and the windbox is controlled by the secondary air shrouds or dampers. Both the amount of secondary air and the vertical distribution of secondary air in the boiler affect NOx emissions (see Figure 9). Biasing the secondary air flow toward the top of the furnace creates a staged combustion affect, which

reduces NOx emissions. The parameter α characterizes the vertical biasing of the shroud settings. Alpha can vary between -1 and 1, where positive values indicate biasing towards the top of the furnace and negative values toward the bottom of the furnace. The equation for α is shown below. In this equation, N is the number of burner elevations, S_j is the average shroud opening for the jth burner. $S_{j,Max}$ and $S_{j,Min}$ are the maximum and minimum shroud openings, respectively. The parameter i is equal to N/2 - 1.

$$\alpha = \frac{\sum_{j=0}^{i-1} \frac{N-1-2j}{2}\left(S_{j+1} - S_{N-j}\right)}{\sum_{j=0}^{i-1} \frac{N-1-2j}{2}\left(S_{j+1,max} - S_{N-j,min}\right)}$$

Since the goal of the advisor is to reduce NOx, it only considers values of alpha greater than zero. Figure 9 shows the effect of biasing the secondary air toward the top burner.

Biasing the secondary air affects local values of the fuel to air ratio, steam temperatures, unburned carbon, CO levels and other parameters. For this reason, the testing advisor must consider these effects when biasing of secondary air is utilized.

Swirl Settings

The register swirl settings can also be adjusted to affect NOx formation. The swirl settings, like the overfire air and economizer oxygen, are coupled with the secondary air shroud opening and distribution. Test data at Chalk point suggest increasing the amount of swirl tends to decrease NOx (see Figure 10). Thus, the test strategy is to impart as much swirl as possible while maintaining good furnace quality and ignition points. This is

not always possible because the flame quality, flame length and ignition point are all very dependent on the swirl settings.

Mills

The last parameter which has been tested and shown to play a factor in NOx formation is the mill loading pattern (see Figure 11). Like the secondary air shroud openings, the coal flow can be biased vertically toward the bottom burners, which creates an overfire air effect, which in turn reduces NOx. Biasing the mill coal flows towards the bottom burners means having more coal flow to the bottom burners than the top burners. A bias parameter, β, was developed to characterize the amount of mill biasing. This mill bias parameter can range from -1 to 1, where negative values indicate biasing toward the bottom of the furnace and positive values to the top. The equation of β, is shown below. N is the number of mills, i is N/2 -1, M_j is the coal flow of mill j.

$$\beta = \frac{\sum_{j=0}^{i=1} \frac{N-1-2j}{2}\left(M_{j+1} - M_{N-j}\right)}{\sum_{j=0}^{N} M_j}$$

Since the goal is to try to create a staged combustion effect, the mills should be biased towards the bottom of the furnace as much as possible. Mill biasing is only possible when excess mill capacity exists. Biasing the mills can have adverse affects on parameters such as steam temperatures, CO emissions, and on LOI.

Uncontrolled Parameters

Although the parameters just described can be measured and controlled, there are some parameters which cannot. One example is boiler cleanliness. Over time, slagging occurs causing a dirty boiler. Slag, which builds up on the water walls, tends to increase the furnace temperatures, which increases NOx. Steam temperatures also usually increase as slagging progresses, which affects unit heat rate. Uncontrolled variations in fuel quality are also important because of their effects on NOx, unburned carbon, and slagging patterns.

To minimize the effects of uncontrollable factors on the test data, baseline tests are performed at the beginning and end of the day. In addition, the advisor allows the user to run baseline tests at other times. Using the baseline tests performed at the beginning and end of each day, the test data for that day are normalized to account for these uncontrollable factors. The normalization determines the average baseline settings for each day, then corrects the NOx for each test point by subtracting the change in NOx for the day.

Experimental Procedure

In developing the test logic in the advisor, it was assumed that most parameters are relatively independent of the others. For these parameters, tests are run in which only one parameter is varied while the others are held fixed. Special tests are then run to account for interdependencies between overfire air setting and O_2 level, and between secondary air shroud setting and swirl setting. In the case of the overfire air / O_2 and secondary air /

swirl test sequences, a factorial test method is used along with a parametric approach. For example, two or three test points of O_2 levels are selected. Then for each O_2 level, several overfire air settings were tested while holding all other parameters fixed. This generates a database in which combinations of overfire air setting and O_2 level are tested, providing data on parameters which depend on one another. The testing sequence is based on which parameters have the largest effects on NOx. Economizer oxygen and overfire air are two of the most important parameters.

When a target NOx is required, an external curve fitting program is used to predict what value of a parameter will produce the target NOx. The external curve fitting program is a DOS application written in C. It performs a linear least squares curve fit on the test data and returns the slope and intercept of the linear best fit line. This technique was used because it predicts the actual trend of the data, while reducing the effects of scatter. It also takes into account the affect of the uncontrolled parameters.

The next section discusses the logic for each testing module and the advisor operation.

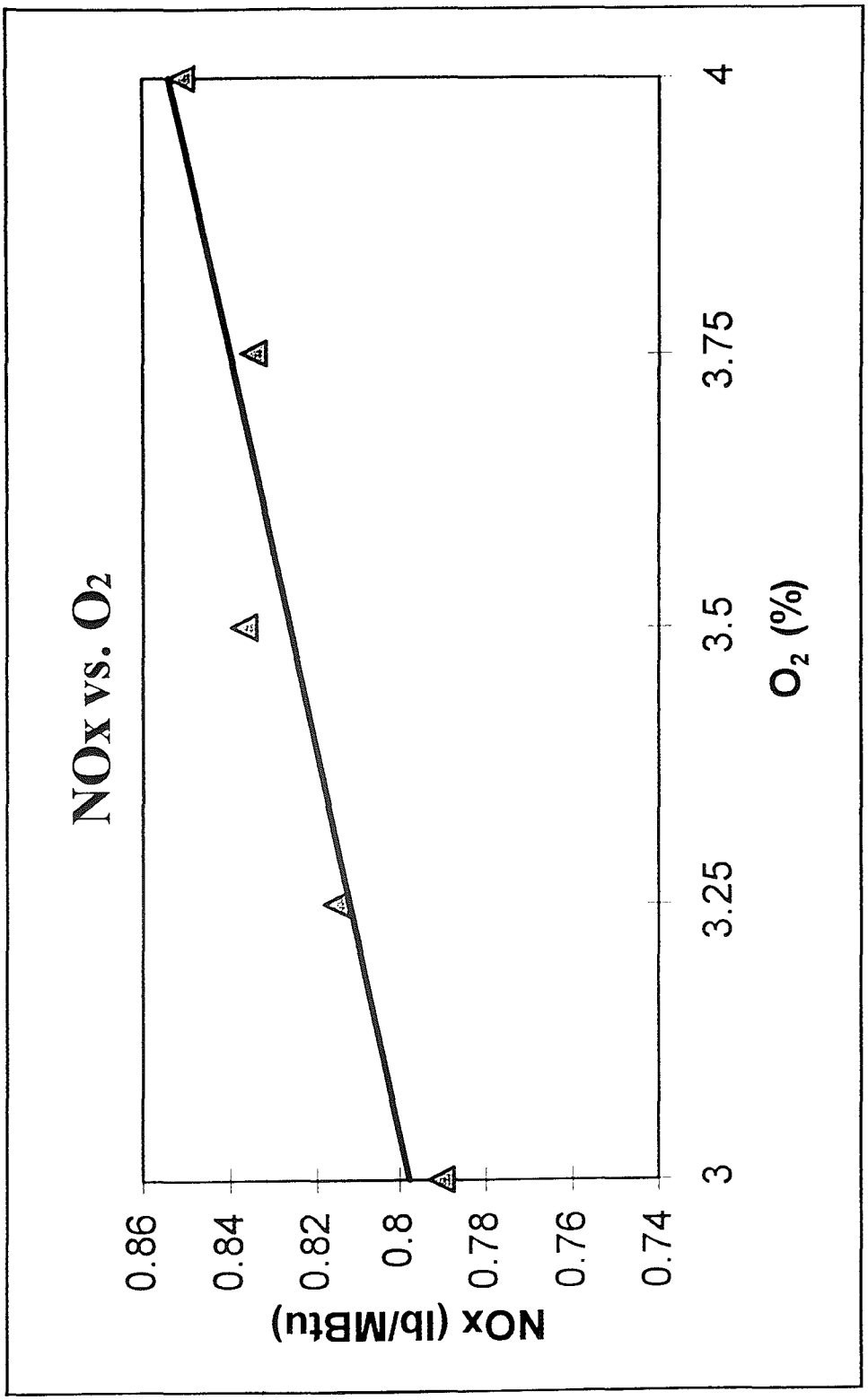

Figure 5 - NOx vs. O_2

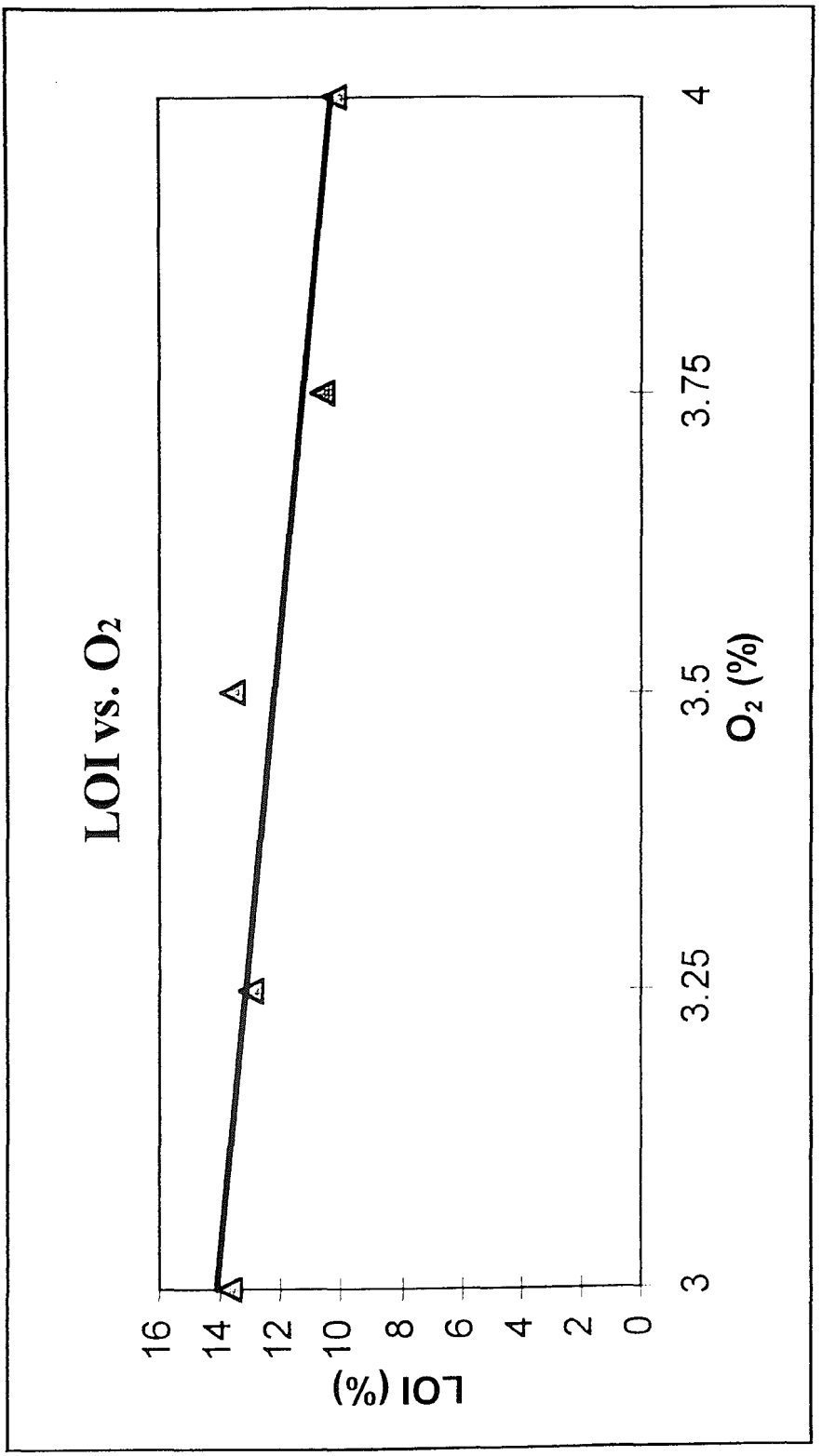

Figure 6 - LOI vs. O_2

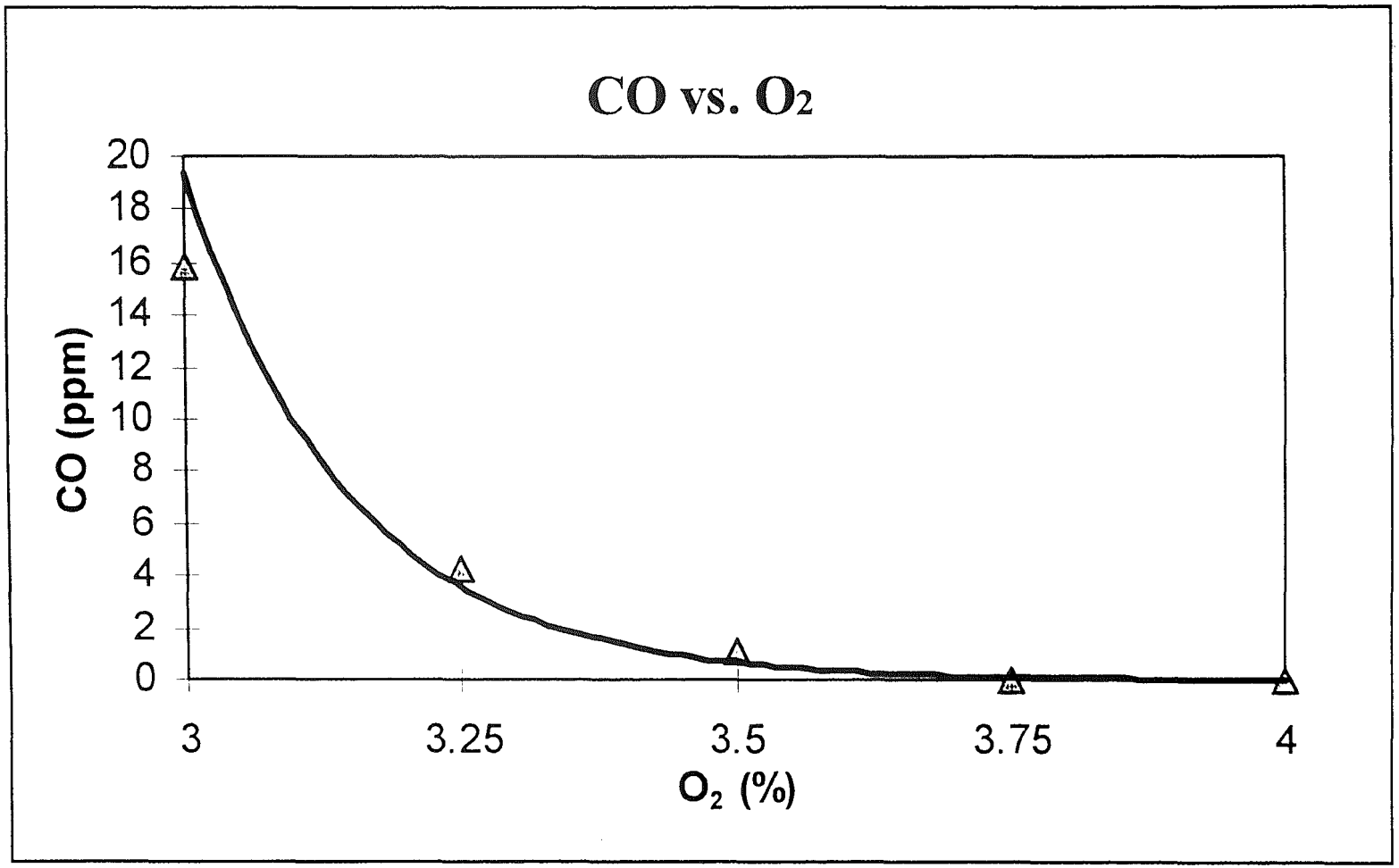

Figure 7 - CO vs. O_2

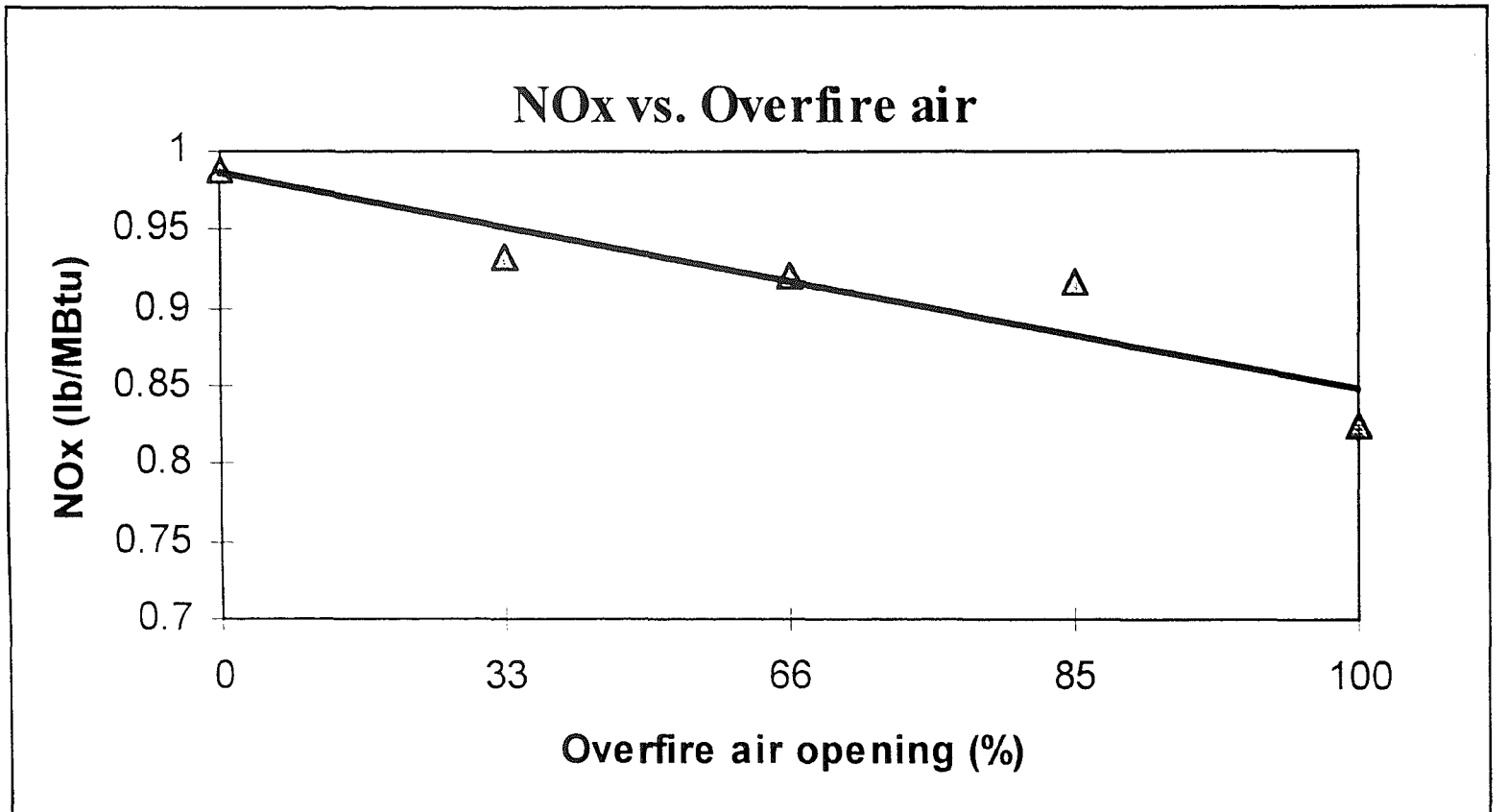

Figure 8 - NOx vs. Overfire Air Opening

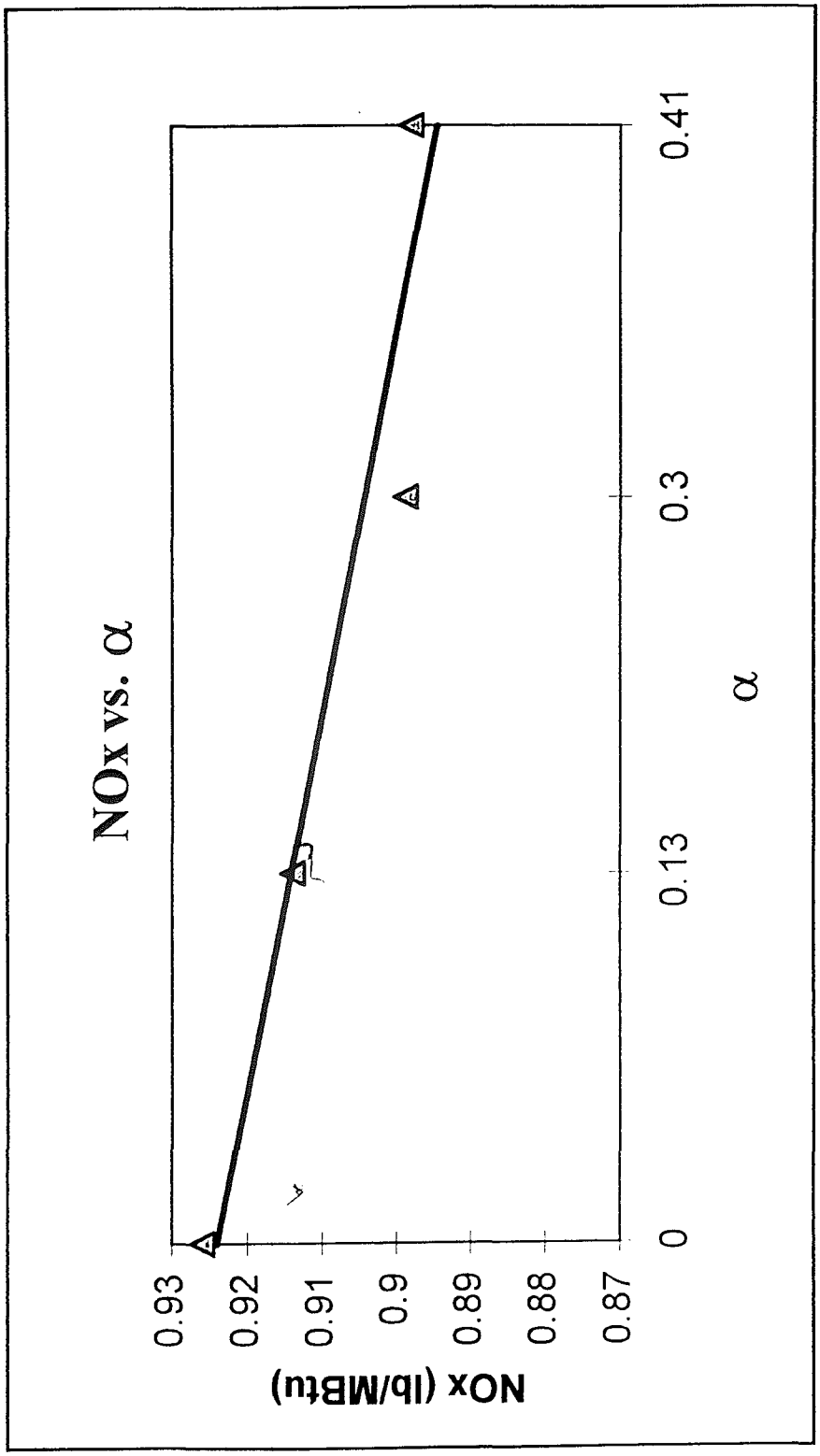

Figure 9 - NOx vs. α

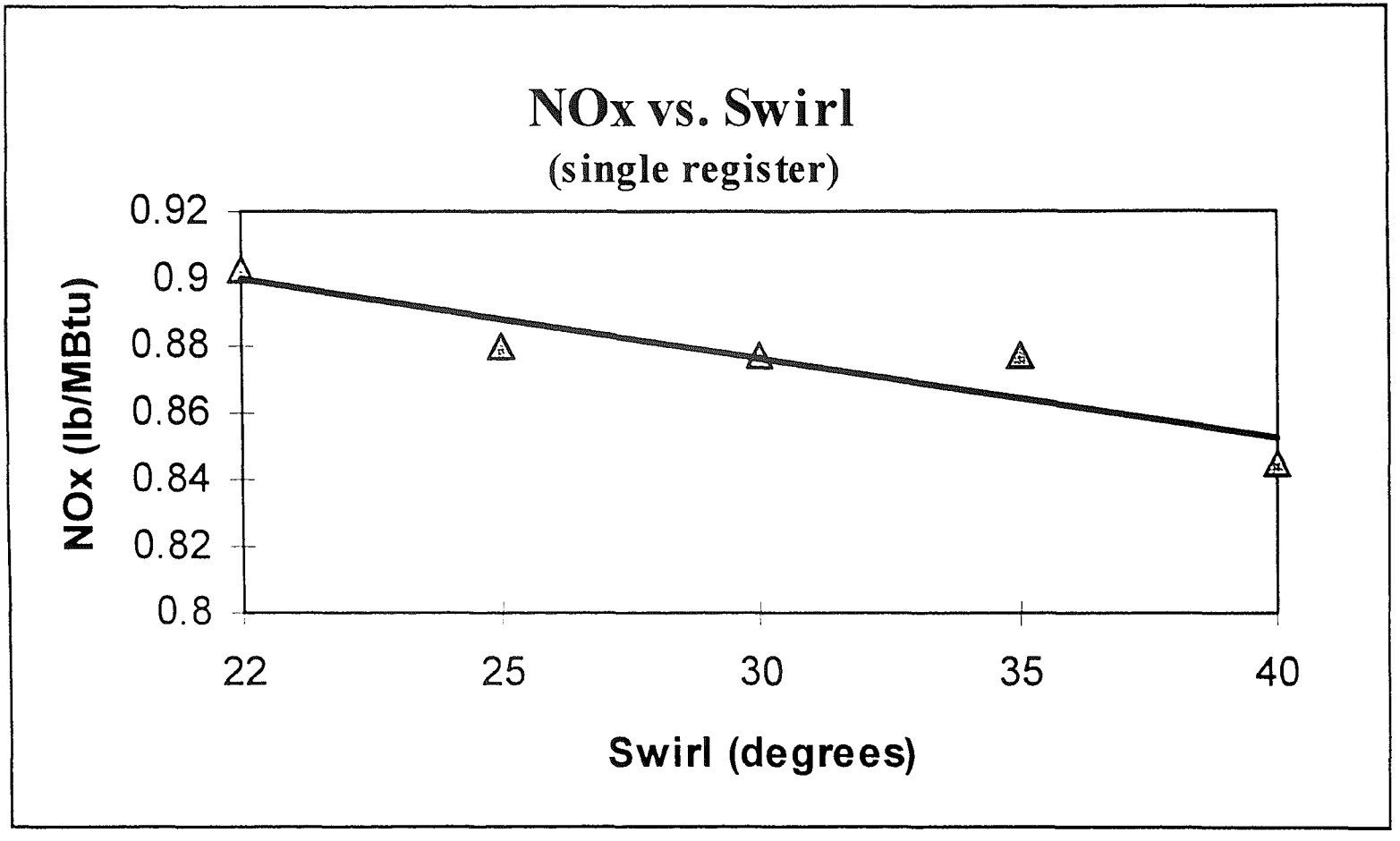

Figure 10 - NOx vs. Swirl settings

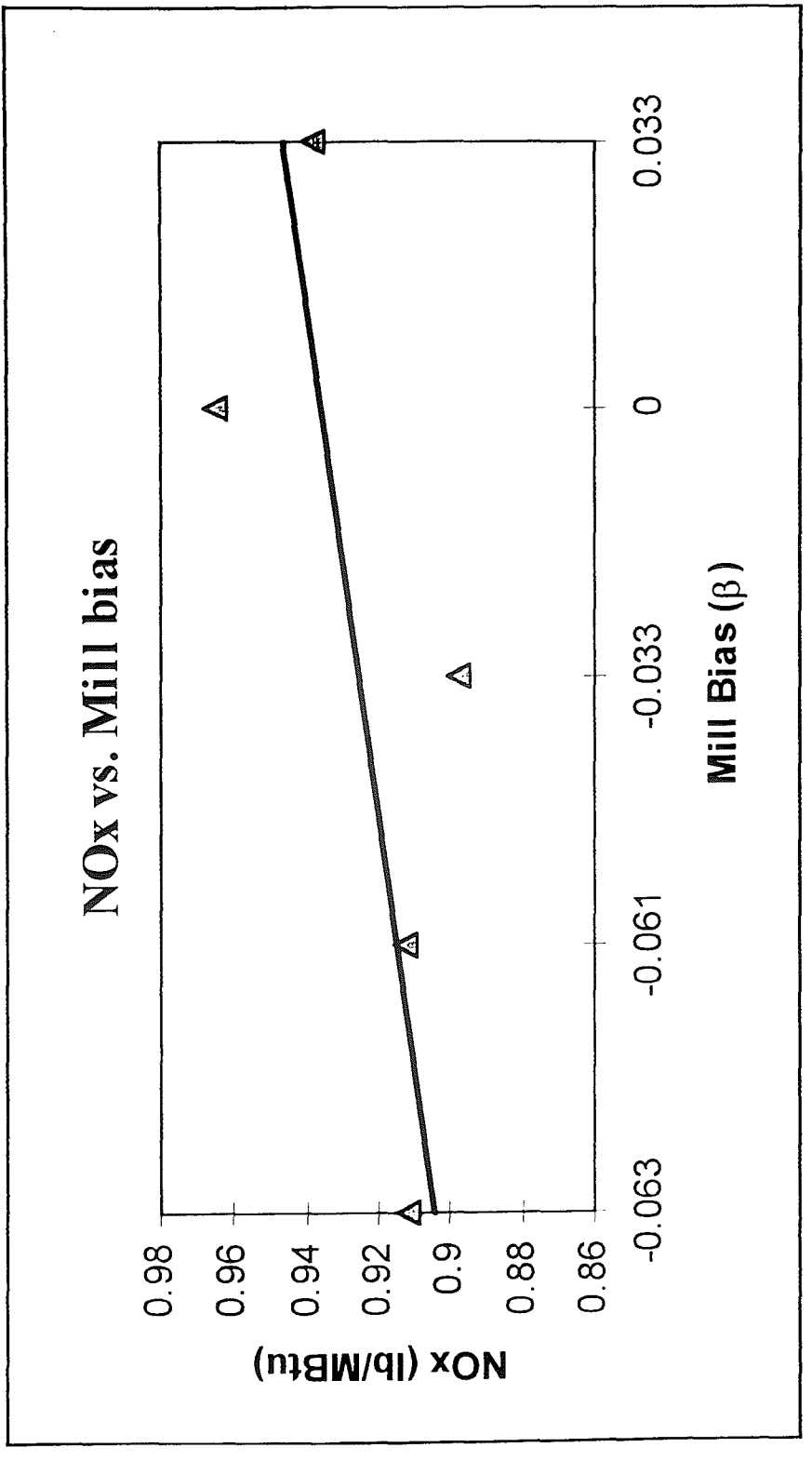

Figure 11 - NOx vs. Mill Bias

Software Description

Overall Operation

The wall-fired boiler testing advisor is designed to operate in two distinct modes. The first mode finds the absolute minimum NOx with which the boiler is able to operate safely. The second mode determines what boiler settings allow the boiler to meet a target NOx while minimizing heat rate with safe boiler operation.

The testing advisor was also designed with features to meet other user needs. The advisor is capable of testing a boiler which has never been tested before and recommends control settings to optimize the boiler. The advisor also has the ability to recommend several short tests for each parameter and then determine which parameters are significant in meeting the goal. This feature can be used to save testing time by not testing parameters which are not significant.

The advisor also has the ability to only test parameters which the user chooses. This allows the user to not test parameters which may have been previously tested or are incapable of being tested. The advisor is also intelligent enough to know if the boiler does not have an overfire air system, then the overfire air / O_2 and the overfire air distribution modules should not be run.

A large portion of the advisor is devoted to the collection and storing of the test data. A significant amount of data is gathered as soon as the advisor is started. These test data are used to determine the next step in the testing procedure. The quantitative data

are obtained directly from the plant monitoring system and entered into the advisor by the user. Qualitative data, such as ignition points and furnace quality, must be determined by the user and entered manually. In the future, a user interface will have the ability to communicate with the plant data acquisition system and collect the data via a modem or direct connection.

The advisor stores data in two ways: facts and objects. Rules are used to alter facts and objects throughout the advisor. All information in the advisor is stored in the form of facts with the exception of the test data, which are stored in objects. An object called ***test-data*** stores the following information for each test point:

- test number
- test date
- Beginning and ending time of the test
- economizer oxygen (%)
- mill flow (klbs/hr)
- mill bias
- shroud openings (%)
- shroud bias parameter (α)
- furnace oxygen (%), if available
- windbox pressure (in H_2O)
- main steam temperature (°F)
- reheat steam temperature (°F)

- opacity (%)
- CO (ppm)
- LOI (%)
- mill amps (amps)
- mill pressures (in H_2O)
- mill exit temperatures (°F)
- main steam spray
- reheat steam spray
- load (MW)
- NOx (lbs / Mbtu)
- furnace quality (good / bad)
- ignition point (good / bad)
- overfire air openings (%)
- overfire air bias (%)
- swirl setting (degrees)

This object is used to pass test data between the ***main*** module and the other testing modules.

The advisor also uses internally defined functions and an external program to process some of the test data. Functions defined in the advisor are used to calculate the mil bias, shroud bias and to process the user's responses. An external program,

v1least.exe, which performs regression analysis is also used. This program uses the least squares method to determine the coefficients for a best fit line.

The main reason for choosing an expert system, as opposed to a conventional programming language to develop the advisor, is the flexibility required in the logic for incorporating new features and new modules. Expert systems are much less structured and procedure oriented than conventional programming languages. Another advantage of expert systems is the ability to see what is in the system's memory in a readable form. This makes it easy to modify and validate expert systems.

The testing advisor consists of the following modules: ***Start-up***, ***Main***, ***significant parameter***, ***economizer-oxygen***, ***overfire air / O_2***, ***secondary air / swirl***, ***overfire air distribution*** and ***mills***. The ***start-up*** module is visible to all other modules, while the ***main*** module is visible to all other modules except the ***start-up*** module (see Figure 12).

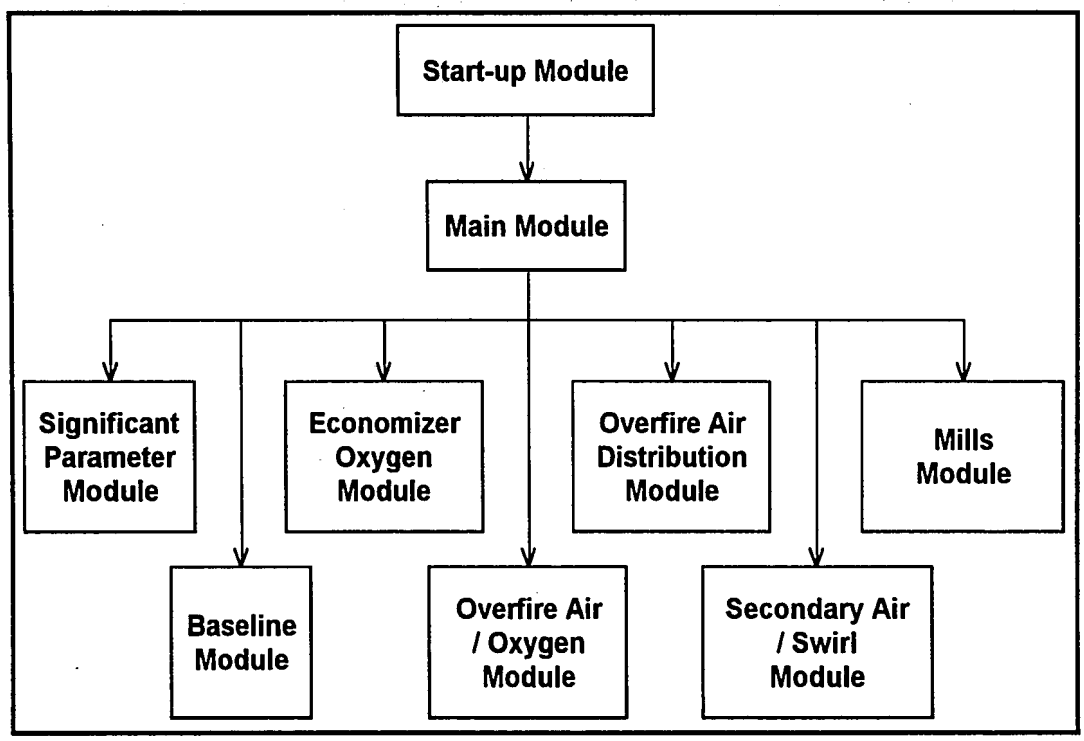

Figure 12 - Visibility of modules

The ***economizer-oxygen***, ***significant parameter***, ***overfire air distribution*** and ***mills*** modules are all based on the parametric test method. The ***overfire air / O2*** and ***secondary air / swirl*** modules are based on a factorial testing method. The ***start-up*** module is the first module to run and obtains information on the boiler and configures the advisor. The ***main*** module is then run and controls the entire testing procedure until the advisor is terminated. Figure 13 below shows the input and output of the advisor.

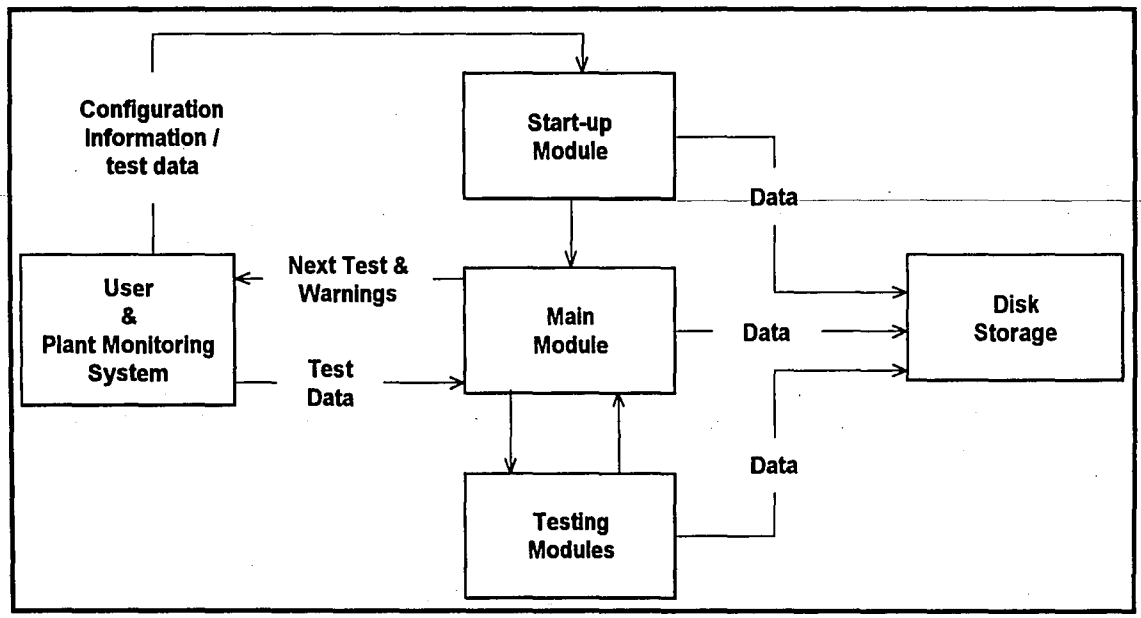

Figure 13 - Input and output of advisor

When the testing advisor is started, the ***start-up*** module takes control to initialize the advisor (Figure 14 shows the flow chart of the advisor). The user is then asked for the goal of the session which is either to find the settings which give minimum NOx, or to find the settings which give a target NOx while minimizing heat rate.

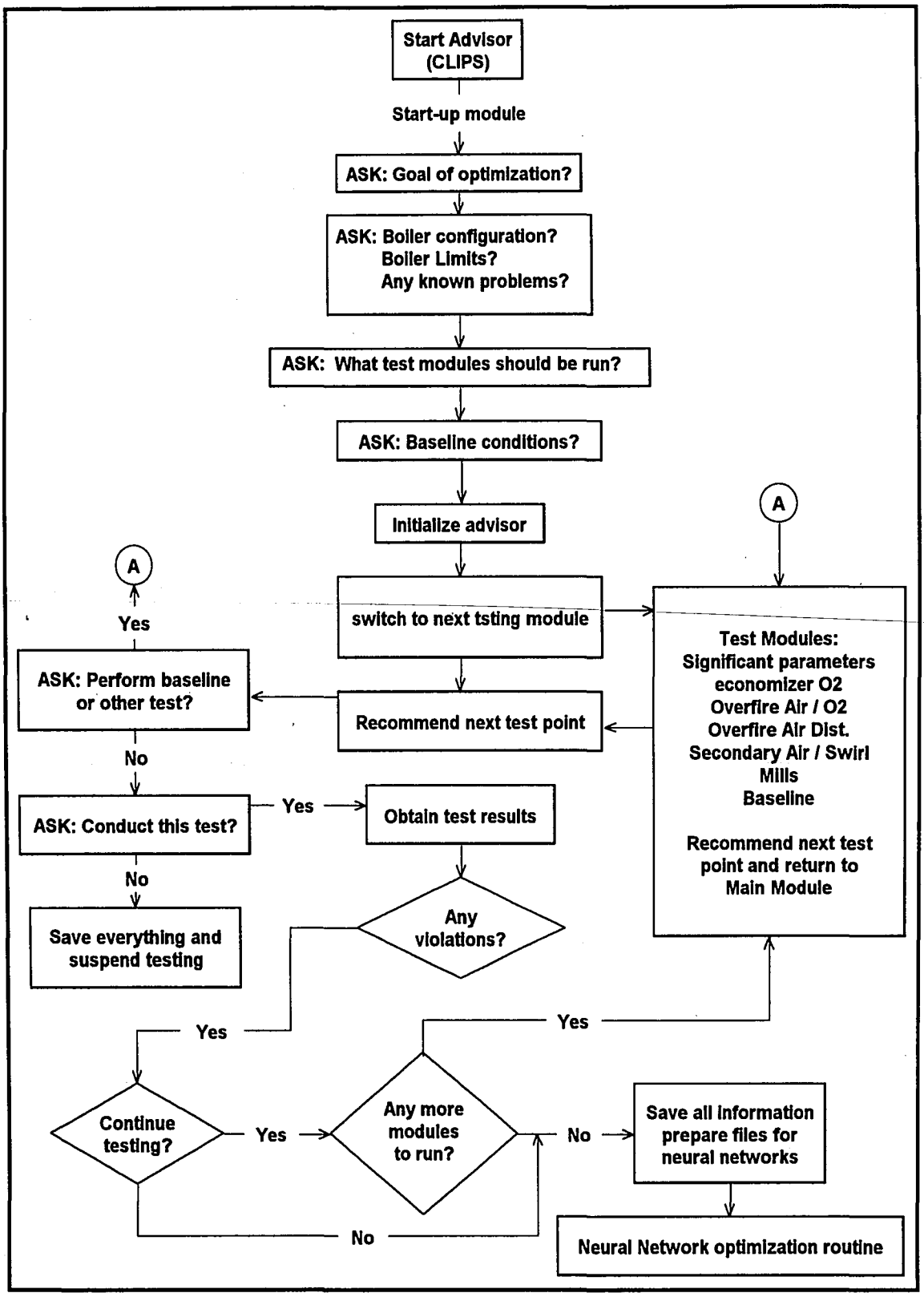

Figure 14 - Flow chart of overall operation of advisor

If the second option is chosen, then the user must enter a target NOx. The user is then asked if any parameters have already been or are incapable of being optimized. If so, those parameters will not be tested. Next, the user is asked if the ***significant parameter*** module should be run. After that, the advisor must be configured for the particular boiler. This is done by asking a series of questions about the boiler, including:

- the number of mills
- the number of burner elevations
- the number of overfire air dampers
- single or opposed unit
- single or dual register burners

Then the user is asked a series of questions regarding boiler safety and configuration limits, including:

- minimum safe furnace O_2
- maximum mill capacity
- minimum and maximum windbox pressure
- minimum and maximum main steam temperatures
- minimum and maximum reheat steam temperatures
- maximum allowed opacity
- maximum allowed LOI
- maximum allowed mill amps
- minimum and maximum mill pressures

- minimum and maximum shroud openings
- minimum and maximum overfire air damper openings
- minimum and maximum swirl settings

This information is then used throughout the testing procedure to assure safe boiler operation.

Next, the user is shown the default modules which will be run and asked to select which modules should be run. The user is then asked to provide the baseline operating settings for the boiler. These baseline settings will be used to run a first baseline test. This completes the task of the ***start-up*** module and the control of the program is then passed to the ***main*** module to begin the testing procedure.

The ***main*** module first determines what module to run, then it passes control to that testing module. The next test module depends on the boiler configuration and which modules the user chooses. The test module then recommends a test point and passes control back to the ***main*** module.

The ***main*** module then notifies the user of the next test point along with instructions on how to conduct the test. Before the test is conducted, the user is given the option of suspending the testing, in which case the status of the program is saved and it is then halted, at a later time, the user is then able to continue the testing exactly where it was suspended. This is an important feature because it allows the testing procedure, which could last two weeks or more, to be conducted at convenient times.

After the test is performed, the advisor obtains the required data from the user. The user must collect the data from the data acquisition system and manually enter them.

The data are then analyzed to check to see if any limit violations have occurred. If a violation occurred the user is notified and asked if the testing should continue or be halted. This is a judgment call and is at the user's discretion. If the user decides to stop the testing, the status of the program is saved and the test sequence is terminated. Otherwise, the test data and any violations are passed to the current testing module. The testing module then determines if more testing is needed. If more testing is needed, the next test point is recommended and passed back to the ***main*** module.

Recommended test points are determined from the current state of the boiler and the knowledge previously discussed. In most circumstances, recommended test points are specific settings, with a few exceptions: the first test of mills and secondary air shrouds is never specifically defined. In this case the objective is to determine the distribution pattern which gives the lowest NOx. Since this is not known, the user must adjust these parameters to their limits in order to have a reference for the remaining mills and shroud tests. In these cases, only loading pattern and test instructions are given to the user.

Once the current testing module has determined no more tests are needed, the data may be analyzed using the ***v1least.exe*** program. This is used to determine the approximate control settings which give the minimum NOx.

If the ***main*** module gets another recommended test, the same procedure just described is repeated. Once the current testing module is complete, the ***main*** module then determines the next module to be run and passes control to it.

Once the advisor determines no more modules remain to be tested, it prepares the database and other files for the neural network / optimization package [2]. The output

files include the goal of the session, the target NOx if required, parameters tested and the parameter limits.

Start-Up Module

The ***start-up*** module performs all initialization and configuration of the advisor for the boiler being tested. It also defines all internal functions and objects used to store the test data.

The functions defined can be classified into two categories: input-output functions used to interface with the user; and mathematical functions. The input-output functions include ***choose_ask***, ***Y_N_ask***, and ***run-window-app***. ***Choose-ask*** prints a question with choices to the screen and gets the answer from the user. ***Y_N_ask*** is similar to ***choose_ask,*** except the only valid responses are yes or no. ***Run-window-app*** uses ***Y_N_ask*** to request the user to run an external program and enter yes once it is completed. All three of these functions print an error message if an incorrect choice is entered. Other defined functions calculate the mill bias using the individual mill settings, and the shroud bias using individual shroud settings.

An object oriented class called ***test-data*** is also defined to store the test data. It is specified as "pattern-match reactive" so rules can use its data just as if it were a fact. It is also defined as "read-write" so rules can both read from the object and write to it. This object holds all the test data discussed previously. Data stored in object are stored in slots. Slots are fields which contain information about the object, and are capable of holding one or more values. If a slot can hold more than one value, then it is called a

multifield slot. There are two other slots called ***status*** and ***par-eval***. ***Status*** is used to indicate what the object is being used for. Valid values include *current*, *previous*, *recommendation*, *optimum1*, *optimum2*, *optimumave*, *baseline* and *system-baseline*. *Current* indicates the object is storing data for the current test in progress. *Previous* indicates the object holds data from a previously run test. *Recommendation* indicates the object holds the next recommended test point. *Optimum1* and *optimum2* hold boliler conditions which give minimum NOx. *Optimumave* holds the average operating conditions of *optimum1* and *optimum2*. *System-baseline* holds the user specified baseline conditions, while *baseline* holds data from a baseline test. Test objects are initialized as *recommendation*, then become *current* and eventually become *previous*. The test database is made up of the data from all instances with the *previous* specifier.

The slot ***par-eval*** is used to indicate what parameter was tested for the test data stored in the instance. Valid values include *eco*O_2, *sig-par*, *OFA*, *sec-air*, *OFAD* and *mills*.

Initial facts to start the module operating and the default testing strategy are first asserted. Asserting means saving information in the form of a fact in the system memory. The first fact asserted assures that the test objective is the first question asked (refer to Figure 15 for flow chart).

A rule called ***user-IO-initialization*** first asks the user for the goal of the session. If the goal is to minimize the heat rate subject to a target NOx, the user is then asked for a

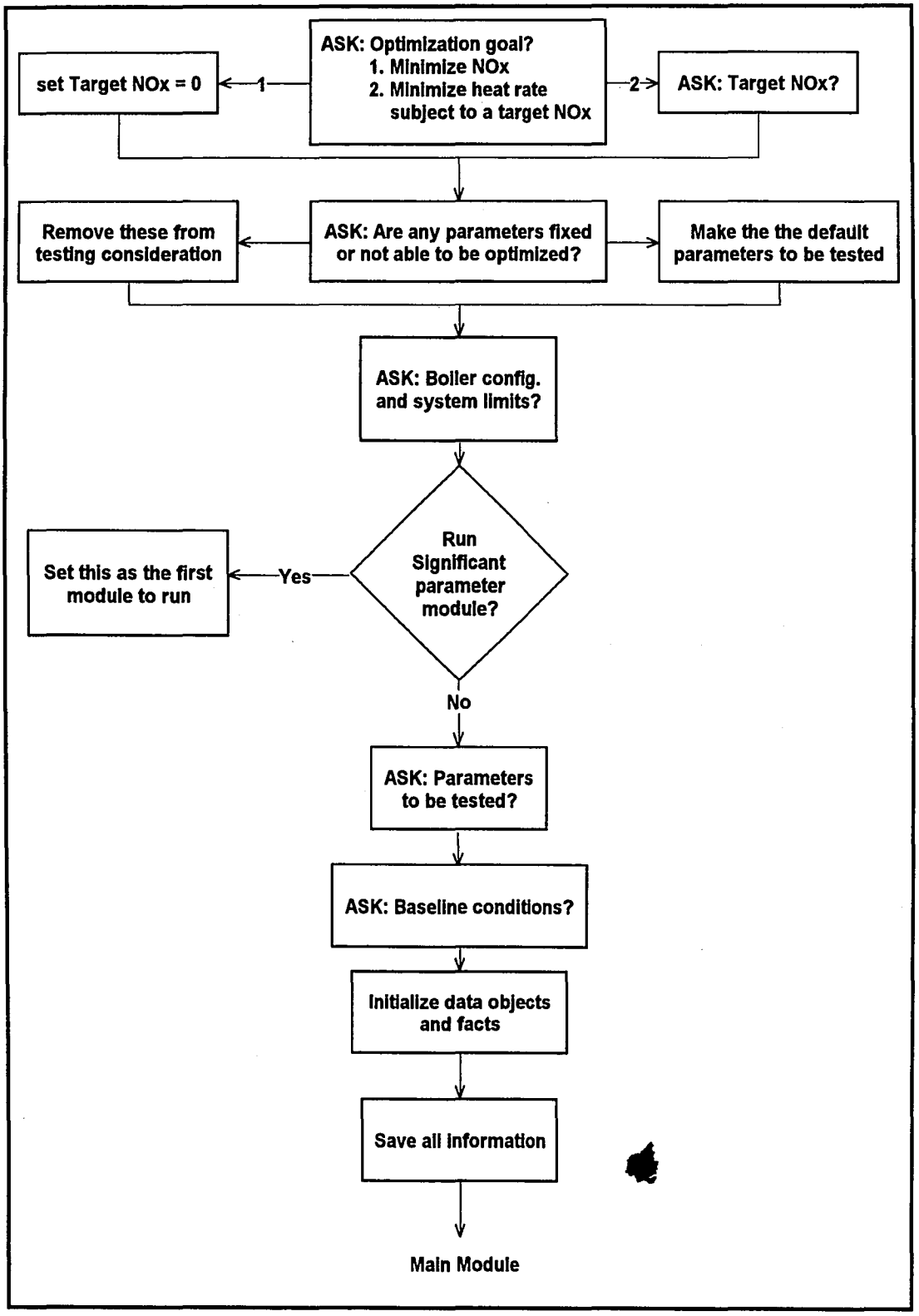

Figure 15 - Start-up module flow chart

target NOx. This information is then stored in facts. Next, the user is asked if any parameter can not be tested, and then stores the response in facts. Then the user is asked if the ***significant parameter*** module should be run. Finally, the same rule prompts the user for the boiler configuration and limits. These are then stored in facts, and then rules use these data to determine some parameter ranges and to eliminate any modules which can not be run. For example, if the boiler does not have an overfire air system, the ***overfire air / O_2*** and ***overfire air distribution*** modules will not be run. The elimination is done by the ***evaluate-sequence*** rule.

Next, rule called ***get-baseline*** prompts the user for the baseline data and stores these data in an object with the ***status*** of *system-baseline.* This information is also stored in the recommended object for initialization reasons. Finally, a rule called ***initial-goal*** is fired and control is passed to the ***main*** module.

Main Module

The ***main*** module is the portion of the testing advisor and controls everything once the ***start-up*** module is finished preparing the advisor for testing. The module starts by asserting a few facts before the program begins executing. Such facts include information about the significant NOx reduction, starting test number and the initial minimum NOx.

A second set of facts, which describe the testing order of the modules, is also asserted before execution. These facts tell the system what the order should be if one or more modules are skipped. These facts include: "If nothing is optimized, then goal is significant parameter tests"; "if significant parameter tests is optimized, then goal is

economizer oxygen"; "if economizer oxygen is optimized, then goal is overfire air / O_2"; "if overfire air O_2 is optimized, then goal is secondary air / swirls"; "if secondary air / swirls is optimized, then goal is overfire air distribution"; "if overfire air distribution is optimized, then goal is mills"; and "if mills is optimized, then goal is finished".

A rule called ***perpetuate*** looks through the facts in the system matching "optimized .. is" and determines what the goal is, or what modules should be run (refer to Figure 16 for flow chart). For example, if the user chooses not to run the ***significant parameter*** and the ***economizer oxygen*** modules, this rule determines the first module to run is ***overfire air / O_2***. A rule called ***change-module*** is then fired and passes control to the appropriate testing module.

Before the control is passed to the testing module, the current state of the ***main*** module is saved. When control is passed back to the ***main*** module, the state is saved again. This means the current facts and objects in the ***main*** module are saved to disk. This is done as a safety measure, so the advisor can be re-started where it left off in case of a power outage. This is also done so the user can deliberately suspend testing and continue at a later date.

The program then comes back to the ***main*** module with the next recommended test point stored in an object with ***status recommend***. Then, the rules ***do-test*** and ***run-the-test*** notify the user of the next test point with instructions on how to conduct the test. The user is also given the option of suspending the test. If testing is not suspended, the rule ***test-in-progress*** conducts the test and the rule ***create-results*** obtains the test data from the user. The rule ***test-results-temporary***, then, prints the test results to the screen and stores

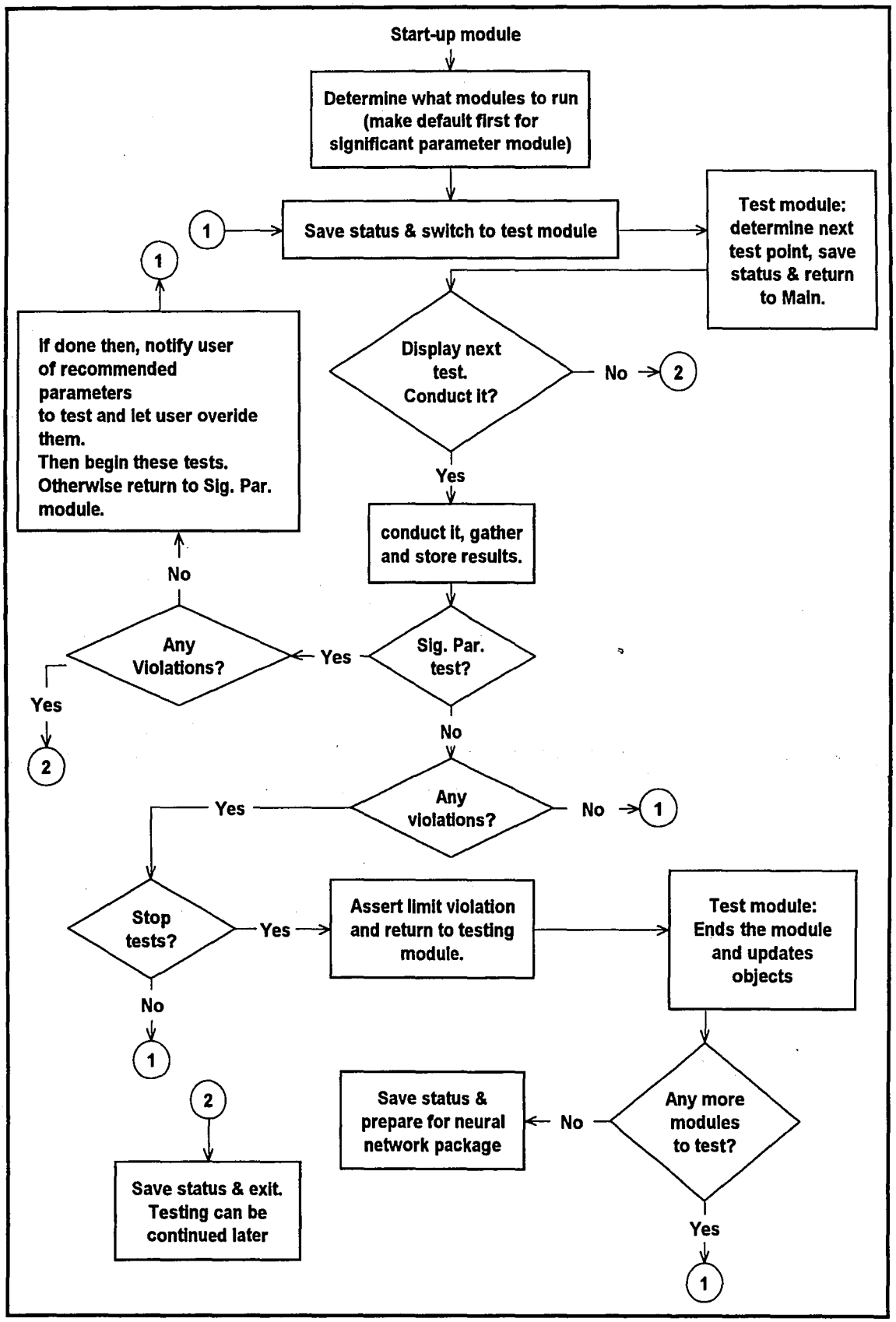

Figure 16 - Main module flow chart

the data in an object with ***status*** *current.* Afer that, a series of rules check the data against the system limits to determine if any violations have occurred.

If a violation has occurred, the user is warned and given the option to either stop testing or continue. If testing is suspended, the rule ***suspend-testing*** saves the current state of the advisor and terminates. If testing continues, control is then passed back to the testing module for the next test point.

A rule ***optimized*** is fired if the testing module has finished and an object with ***status*** *recommend* and ***par-eval*** *optimized-par* exists. This rule asserts a fact indicating the testing module is finished. It also prints the best control settings for that parameter to the screen. Once all the recommended test modules have completed, the rules called ***print-optimum*** and ***nn-switches*** are fired. These rules print the best control settings for each parameter to the screen and prepare the input files for the neural network / optimization routine. If other testing modules still remain to be run, the ***perpetuate*** rule is fired again and the entire process repeats.

Significant Parameter Module

The ***significant parameter*** module is used to test several parameters to determine which one may not be significant in reaching the goal. This is done by eliminating any parameters which do not reduce NOx levels by an amount at least equal to the significant NOx reduction.

The ***significant parameter*** module has initial facts in its knowledge base indicating the parameters to be tested, their increments and their starting values. The starting

economizer oxygen level is 4% and its increment is 1%. The minimum overfire air damper setting is also initialized to 30%. This module tests only the economizer oxygen parameter, overfire air parameter, secondary air shrouds and the mill distribution.

The ***test-what*** rule is first fired to determine if all four of these parameters can be tested (refer to Figure 17 for the flow chart). If the user chooses not to test one of these parameters or if it is incapable of being tested, the ***significant parameter*** module would not test it either. The ***test-what*** rule then asserts facts stating what parameters will be tested.

The ***first-test*** rule modifies the recommended instance with the baseline data and updates the ***par-eval*** slot with *sig-par*. Control is then passed to the ***main*** module to perform the baseline test. Once control returns, the object is modified to be *previous* and a fact stating the baseline NOx is asserted. Next, the ***do-o2-test*** rule is fired and modifies the recommended instance with a new economizer level equal to the difference between baseline O_2 and the O_2 increment. Control is then passed back to the ***main*** module to conduct the test. Once control is returned the object is updated to be previous.

Next, the ***do-OFA-test*** rule is fired and modifies the recommended instance with new overfire air damper settings equal to 100%. A fact keeping track of the number of the test is also asserted. Control is then passed back to the ***main*** module to conduct the test. Once control is returned the object is updated to be previous. The ***do-OFA-test*** rule is fired for a second time and modifies the recommended instance with a new overfire air damper setting equal to 40%. The test is conducted and the objects are updated.

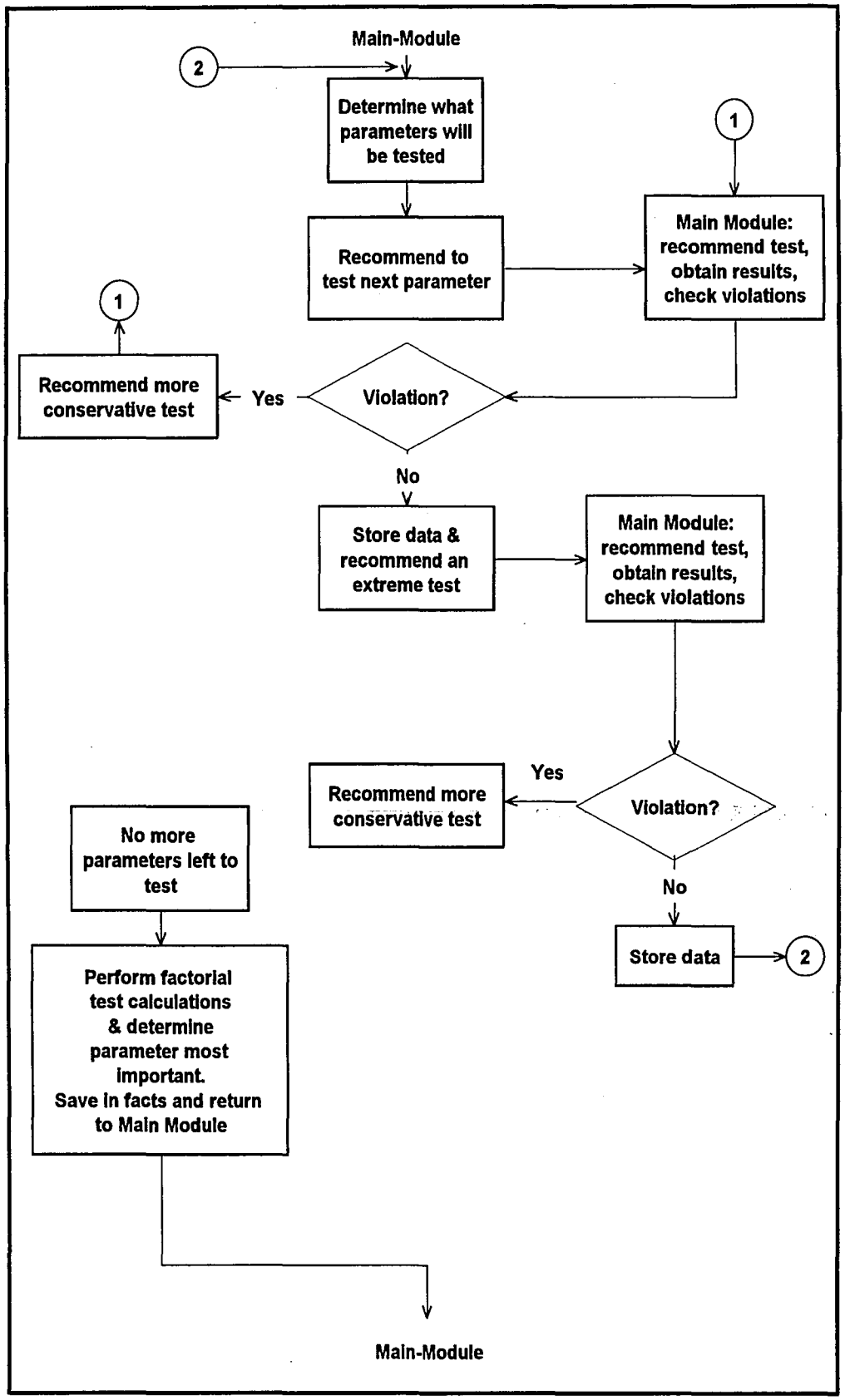

Figure 17 - Significant parameter module flow chart

The ***do-shroud-test*** rule is fired and notifies the user to bias the shrouds as much as possible toward the top. No specific shroud settings are recommended because no reasonable setting is known. The user is also notified to ensure good furnace quality and ignition points are maintained when setting the shrouds. Control is then passed back to the ***main*** module to conduct the test. Once control is returned, the object is updated to be *previous*. Finally, the ***do-mill-test*** rule is fired and notifies the user to bias the mills as much as possible toward the bottom. Just as for the shroud settings, no specific mill settings are recommended because no reasonable settings are known. The user is also notified to maintain safe boiler operation when adjusting the mill loading pattern. Control is then passed back to the ***main*** module to conduct the test. Once control is returned, the object is updated to be *previous* and testing should be complete. Once all of the tests have been run, the ***find-significant*** rule determines which of the parameters, if any, are not significant. If any parameters are found to be insignificant, they are removed from the list of modules to be tested by the ***main*** module.

During the testing, if any violation occurs, the module recommends to the user to conduct a more conservative test. In some cases, such as for economizer oxygen and overfire air, specific tests are recommended as more conservative. If two violations occur, the module marks that parameter as not significant because the parameter range is too small.

Economizer Oxygen Module

The ***economizer-oxygen*** module has an initial fact in its knowledge base stating that the minimum increment of O_2 is 0.2%. This is the smallest possible increment of economizer-oxygen which is considered to affect NOx.

The rule called ***start-test*** is the first rule fired once control is passed to the ***economizer-oxygen*** module (refer to Figure 18). This rule first determines and sets the average overfire air damper openings. It also determines the first O_2 increment and the new recommended economizer-oxygen level. If the increment is less than the minimum increment, it is set to be the minimum increment. A data file called *o2.dat* is also created and a "1" is written to this file. This file stores the test data for this module. It stores the economizer-oxygen and the corresponding NOx for each successful test. This file is used by the external program ***v1least.exe*** to perform a linear curve fit of the data. Finally, the ***par-eval*** slot is modified to hold *eco*O_2 which indicates parametric tests on economizer-oxygen are being conducted. Control is then passed to the ***main*** module to conduct the test.

When control is returned from the ***main*** module and no limit violations occur, the ***retrieve-data*** rule is fired. This rule modifies the *current* object to be *previous* and updates the test number. The minimum NOx achieved this far and its corresponding economizer-oxygen level are also stored in a fact. Lastly, the economizer-oxygen and NOx from the previous test are written to the *o2.dat* file.

The ***prep-experi*** rule is then fired to prepare for the next recommended test point. Again, a new increment and the new O_2 level are determined and placed in the *recommend* object. Control is then passed to ***main*** module.

This process of recommending a test point, conducting the test, and retrieving the data continues until a limit violation occurs, or the economizer-oxygen level is less than the minimum increment.

Once testing cannot continue, due to one of the reasons stated above, the ***finish-experi*** rule is fired. This rule modifies the *current* object to be *previous* and increments the test number in the *recommend* object. It also modifies the ***par-eval*** slot to hold *optimized-par* which indicates the economizer-oxygen parameter has been tested. If the goal of the session is to minimize NOx, then the ***modify-optimum*** rule is fired, otherwise the ***curve-fit*** rule is fired.

The ***curve-fit*** rule first determines if at least two successful tests have been completed. If not, the rule ends and the ***modify-optimum*** rule is fired. If at least two successful tests have been completed, the rule first copies the *o2.dat* file to *v1data.dat*, which is the file expected by the ***v1least.exe*** program. The user is then instructed to run this program and answer "yes" when it is finished. This program then writes the coefficients of a best fit line to the *v1least.dat* file. Using linear extrapolation, the rule determines the approximate economizer-oxygen level to achieve the target NOx level. The ***modify-optimum*** rule is fired. This rule uses the fact containing the best economizer-oxygen and the corresponding NOx to update the *optimum1*, *optimum2* and *optimumave* objects. After this rule ends, control is returned to the ***main*** module.

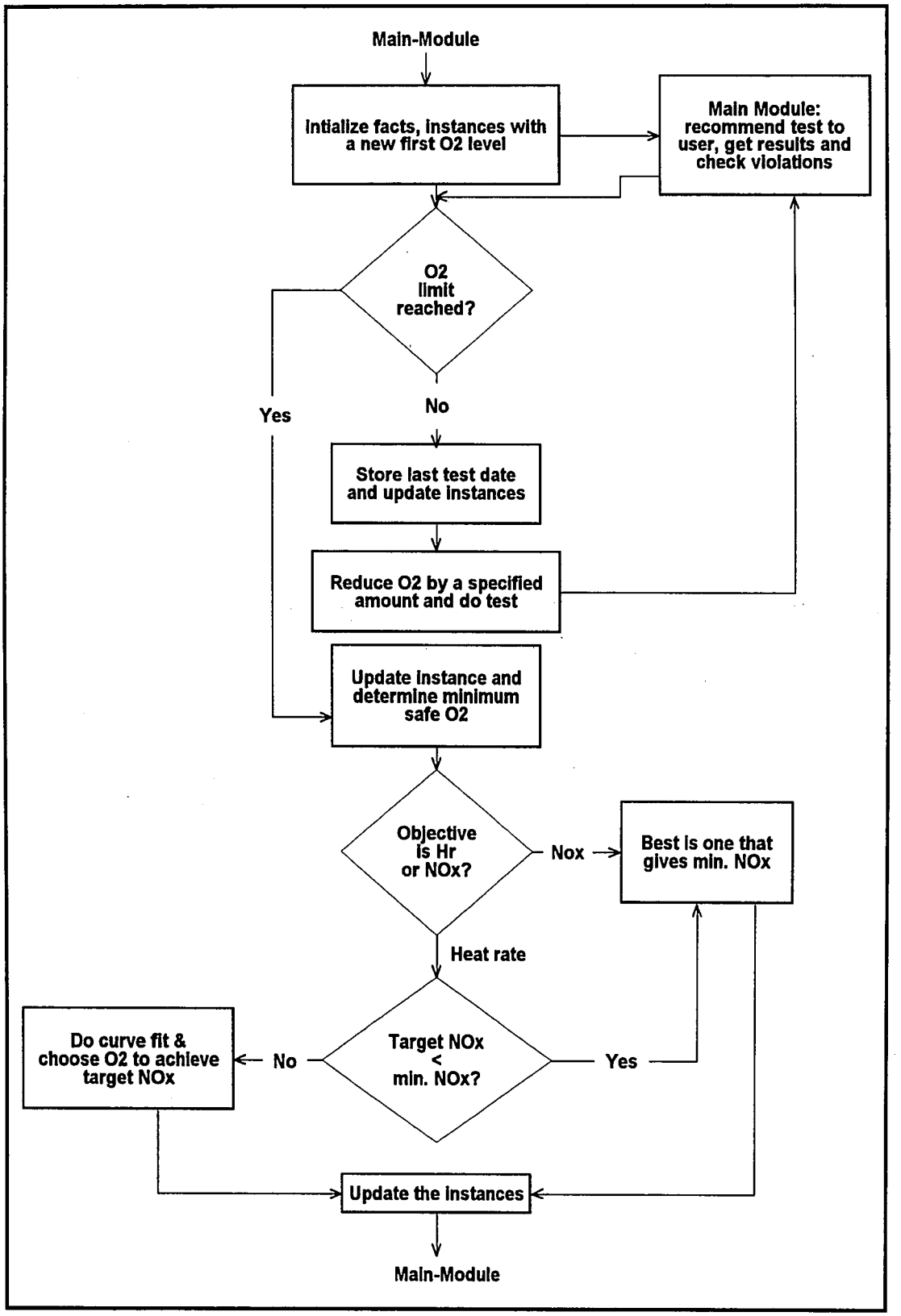

Figure 18 - Economizer-oxygen module flow chart

Overfire Air / O_2 Module

The ***overfire air / O_2*** module has initial facts in its knowledge base, indicating the overfire air damper increment used in forward and reverse tests. Once control is passed to the ***overfire air / O_2*** module, the ***get-constraints*** rule is fired (refer to Figure 19 for flow chart). This rule reads any overfire air damper constraints from a file called *ofacon.dat.* These constraints prevent one or more overfire air dampers from being adjusted. This allows testing of the overfire air dampers, while holding one or more fixed. The constraints are stored in a fact and remain in memory until the module is completed.

Next, the ***start-first-ofa-test*** rule is fired. This rule first places the best economizer-oxygen level from the *optimumave* object and copies it to the *recommend* object. This is the reference O_2 level from which the first overfire air test is run. Next, the ***par-eval*** slot in the *recommend* object is modified to hold the value *OFA* which indicates overfire air / O_2 tests are being conducted. Then, a file called *ofa.dat* is created and a "1" is written to it. The "1" indicates a linear curve fit on the data is to be performed. This file also contains all overfire air / O_2 test data. The initial overfire air damper openings are then determined and stored in the *recommend* object. Then, the first O_2 level is set to be the minimum safe level plus an additional 0.2% and is placed in the *recommend* object. This series of tests consists of a maximum of three tests at this O_2 level while closing the overfire air dampers. Finally, the ***start-first-ofa-test*** rule increments the series number and asserts a fact to count the number O_2 level being tested. Control is thenpassed to the ***main*** module to conduct the test.

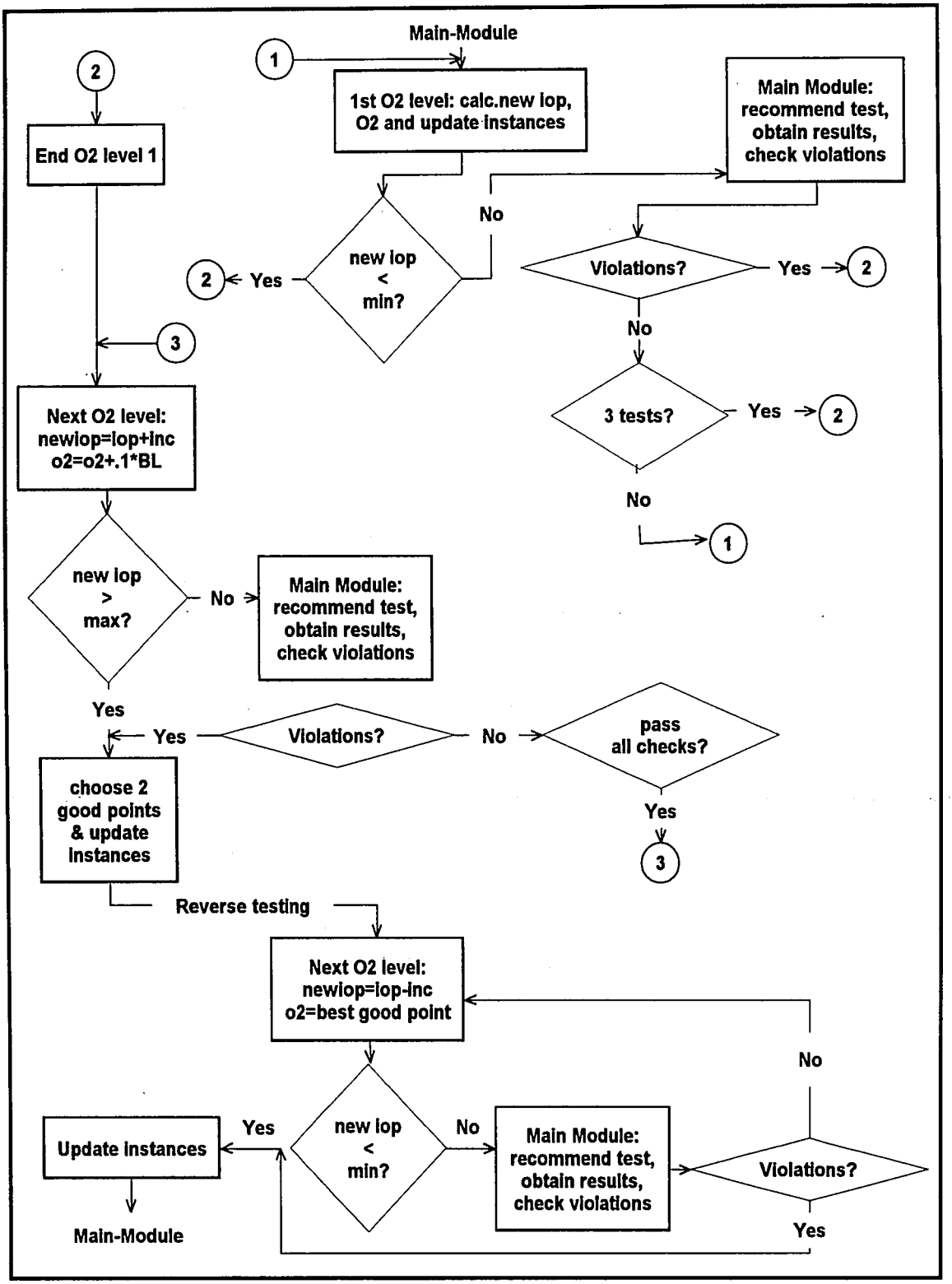

Figure 19 - Overfire air / O_2 module flow chart

After control has returned to the ***overfire air / O_2*** module, with no limit violations occurring, the ***retrieve-data*** rule is fired. This rule is always the first rule to be fired once control is returned to the module and no violations have occurred. This rule modifies the *current* object to be *previous* and updates the test number. The minimum NOx achieved this far, its corresponding overfire air damper settings, economizer oxygen level, and the test level number are stored in a fact. Next, the overfire air damper openings and NOx from the previous test are written to the *ofa.dat* file. Then, the data from the previous test are copied to the *last-test* object. The ***retrieve-data*** rule then determines the next rule to be fired and asserts a fact stating the rule to be fired. If no significant change in NOx is achieved, a fact indicating this is asserted and the appropriate rule will be fired.

If a limit violation did occur during the first series of tests, the ***violation-level-1*** rule is fired. This rule modifies the *current* object to be *previous* and updates the test number. It also asserts a fact indicating the first series of overfire air / O_2 tests is complete, and the second series is in progress. This fact indicates the ***begin-first-test-next-level*** rule is to be fired next.

Once the ***retrieve-data*** rule has fired and the current series number or O_2 level is "1," the ***continue-level-1*** rule is fired. This rule first decreases the average overfire air damper opening by the increment indicated in the initial facts. The new recommended damper openings are then placed in the *recommend* object. The economizer-oxygen level is held fixed at the value used for the first test. Next, the counter keeping track of the number of tests for this O_2 level is incremented by one and control is passed to the ***main*** module.

If a maximum of three successful tests have been completed at the first O_2 level, the ***end-level-1*** rule is fired after ***retrieve-data***. The ***continue-level-1*** rule asserts a fact indicating the first series of overfire air / O_2 test is completed, and the second series is in progress. This fact indicates the ***begin-first-test-next-level*** rule is to be fired next.

Once the previous economizer-oxygen level or overfire air damper series has ended, the ***begin-first-test-next-level*** rule is fired. This rule is used to begin a new series of tests at a higher O_2 level, while opening the overfire air dampers. First, a fact stating the series number and the current O_2 level is asserted. The new O_2 level is then set to be equal to the previous O_2 level plus an additional 0.2%, and it is stored in the *recommend* object. The new overfire air damper openings are then determined by adding the default increment to the current average value. These new settings are also placed in the *recommend* object, and control is returned to the ***main*** module.

Once the ***retrieve-data*** rule has fired and the current series number or O_2 level is greater than one, the ***continue-level-2-N*** rule is fired. This rule first increases the average overfire air damper opening by the increment indicated in the initial facts. The new recommended damper openings are then placed in the *recommend* object. The economizer-oxygen level is held fixed at the value used in the previous test. Next, the counter keeping track of the number of tests for this O_2 level is incremented by one, and control is passed to the ***main*** module to conduct this test.

If a limit violation occurs during the series of tests other than the first series, or a significant NOx change was not achieved, the ***end-level-2-N*** rule is fired. This rule modifies the *current* object to be *previous* and updates the test number. It also asserts a

fact indicating the current series of overfire air / O_2 tests is complete. If the previous two tests did not cause any significant NOx change, no more testing will be performed in which dampers are opened. If more testing is to be done, a fact stating the ***begin-first-test-next-level*** rule is to be fired next. If no more testing in which dampers are opened, a fact to check for reverse testing and a fact to find the best points are asserted.

When testing is complete while opening the dampers, the two points which give the best combination of economizer-oxygen level and average overfire air damper opening must be chosen. If only one O_2 level was run, the ***find-best-1*** rule is fired. If two or more O_2 levels were tested, the ***find-best-N*** rule is fired.

The ***find-best-1*** rule modifies the *optimum1* object to hold the O_2 level and overfire air openings found in the *system-baseline* object. The *optimum2* object is modified to hold the first O_2 level and the maximum tested overfire air settings. The *optimumave* object is modified to hold the average of *optimum1* and *optimum2*.

The ***find-best-N*** rule modifies the *optimum1* object to hold the O_2 level and the tested overfire air openings, which produced the minimum or target NOx. The *optimum2* object is modified to hold the O_2 level and the tested overfire air openings which produced the second lowest minimum or target NOx. The *optimumave* object is modified to hold the average of *optimum1* and *optimum2*.

If more than one O_2 level was tested, a maximum of three reverse tests will be conducted. The rule called ***start-reverse-tests*** is fired first. This rule modifies the *recommend* object to hold the previous tested O_2 level at which minimum NOx was produced. It also determines the starting overfire air settings to be previous settings

tested minus 4%. A fact counting the number of reverse tests performed is also stored. Control is then returned to the ***main*** module.

Once control returns and the ***retrieve-data*** rule is fired, the ***continue-reverse-tests*** rule is fired. This rule decrements the average overfire air settings by 10% and updates the test counter. Then, control is returned to the ***main*** module.

Once a maximum of three reverse tests have been completed, or a limit violation has occurred, the ***end-reverse-tests*** rule is fired. This rule modifies the *current* object to be *previous*. It also modifies the ***par-eval*** slot to hold *optimized-par* to indicate the overfire air / O_2 module has been run.

Secondary Air Module

The ***secondary air*** module has several initial facts in its knowledge base. These facts indicate the swirl increments to be used for single and dual register burners and the swirl increments to be used if limit violations occur during testing.

When control is passed to this module for the first time, the ***get-constraints*** rule is fired. This rule reads in any constraints for the shrouds and single or dual registers. This allows one or more of these settings to be held fixed throughout the testing.

Next, the ***do-baseline-shroud-test*** rule is fired (refer to Figures 20 and 21 for flow chart). This rule first asserts a fact indicating if the tests will be performed using the values from the *optimum1* object or *optimum2* object. Next, the economizer-oxygen level and overfire air damper settings from either *optimum1* or *optimum2* are copied to the *recommend* object. The ***par-eval*** slot in the *recommend* object is modified to hold the

value *sec-air* to indicate secondary air tests are being conducted. Then, a file called *secair.dat* is created and a "1" is written to it. The "1" indicates a linear curve fit on the data is to be performed. This file also holds all secondary air test data. Finally, a fact is asserted to trigger swirl testing once the baseline test is complete. Control is then passed to the ***main*** module to conduct the test.

After control has returned to the ***secondary air*** module, with no limit violations occurring, the ***retrieve-data*** rule is fired. This rule is always the first rule to be fired once control is returned to the module, and no violations have occurred. This rule modifies the *current* object to be *previous* and updates the test number. The minimum NOx achieved this far, its corresponding shroud settings, and swirl settings are stored in a fact and are written to the *secair.dat* file. The current shroud bias parameter, α, is calculated and written to the file. Next, the data from the previous test are copied to the *last-test* object. Finally, the ***retrieve-data rule*** determines the next rule to be fired and asserts it as a fact.

Once the ***retrieve-data*** rule has fired, swirl tests at the current shroud settings will be conducted. After that, the rule called ***first-swirl-test*** is fired. This rule copies the previous shroud settings from the baseline shroud test to the *recommend* object. It then determines the next recommended swirl settings. If the burner has only single registers, the average swirl settings are decreased by a specified amount in the initial facts. If the burner has dual registers, the average outer and average inner swirls are decreased by a specified amount in the initial facts. A fact is then asserted to count the number of the swirl test. Next, the new swirl settings are then placed in the *recommend* object and control is returned to the ***main*** module.

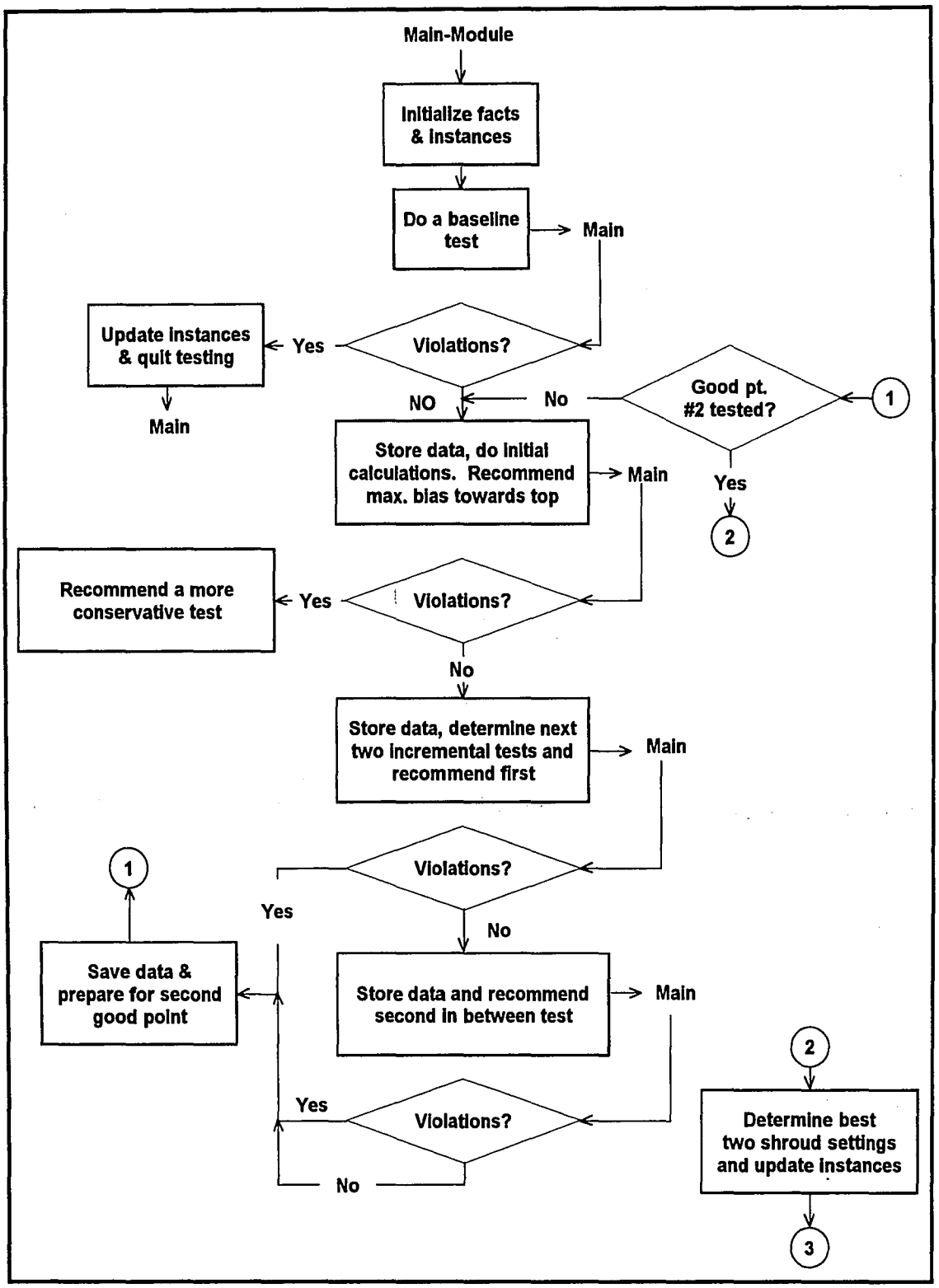

Figure 20 - Secondary air module flow chart

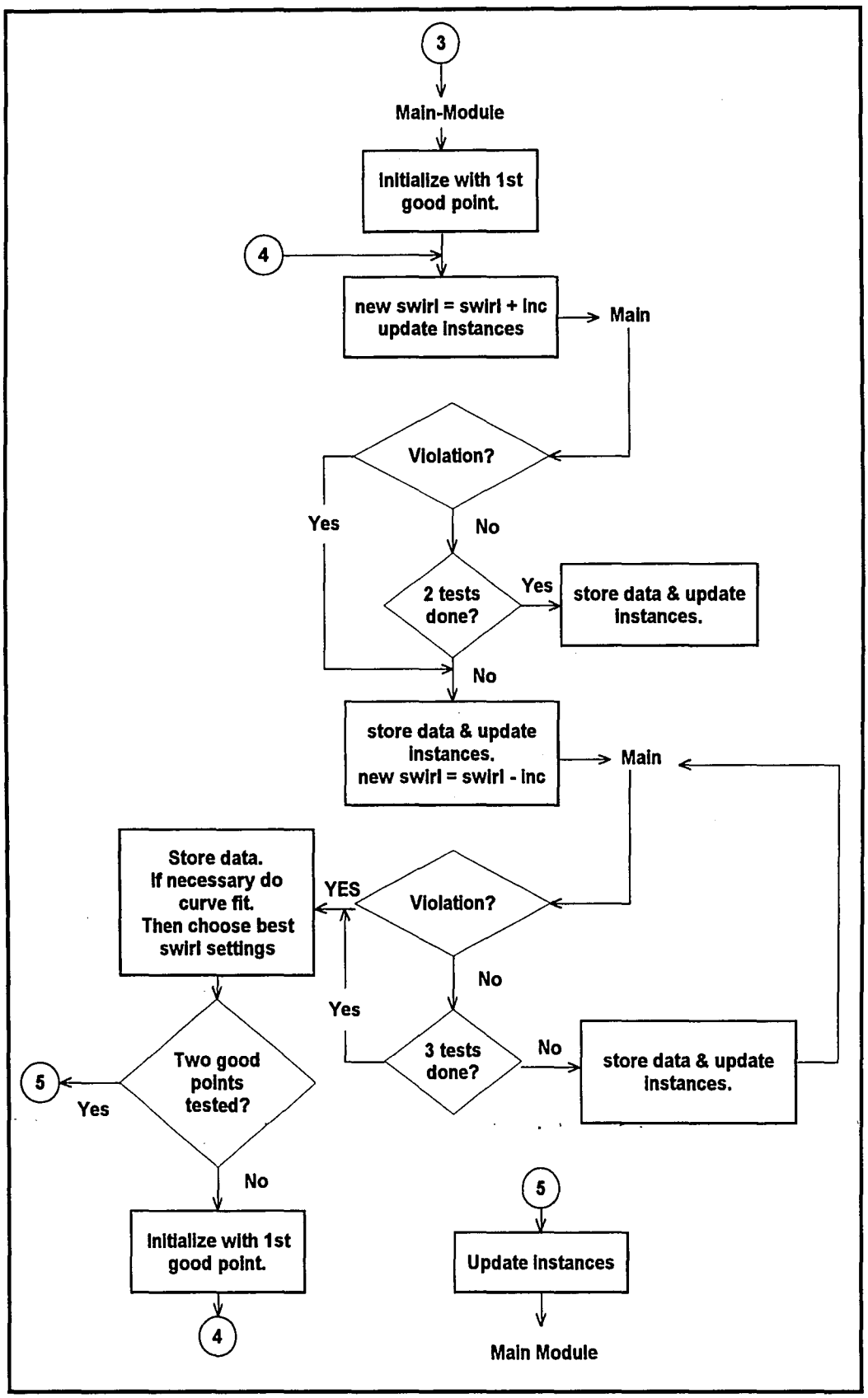

Figure 21 - Secondary air module flow chart (continued)

If a limit violation occurs during the first swirl test, the ***swirl-violation*** rule is fired. This rule modifies the *current* object to be *previous,* and asserts a fact to re-run the previous swirl rule if it was the first violation. If a second consecutive limit violation occurs during a swirl test, the swirl test is skipped and the next test continues.

Once the first swirl tests are complete, the rule called ***second-swirl-test*** is fired. This rule copies the previous shroud settings from the baseline shroud test to the *recommend* object. Then, it determines the next recommended swirl settings. If the burner has a single register, the average swirl settings are increased by a specified amount in the initial facts. If the burner has dual registers, the average outer and average inner swirls are increased by a specified amount in the initial facts. A fact is then updated to count the number of the swirl test. Next, the new swirl settings are placed in the *recommend* object and control is returned to the ***main*** module.

Once the second swirl tests are complete, the rule called ***third-swirl-test*** is fired. This rule copies the previous shroud settings from the baseline shroud test to the *recommend* object. It then determines the next recommended swirl settings. If the burner has a single register, the average swirl settings are increased by a specified amount in the initial facts. If the burner has dual registers, the average outer and average inner swirls are increased by a specified amount in the initial facts. A fact is updated to count the number of the swirl test. The new swirl settings are then placed in the *recommend* object and control is returned to the ***main*** module.

If the burner has dual registers and the third swirl test is complete, the ***second-swirl-test*** rule and ***second-swirl-test*** rule are run again, but this time the outer swirl settings are increased by a specified amount from their previous value.

Once the swirl tests are complete for the baseline shroud settings, a second shroud test will be run. The ***second-shroud-test*** rule is fired and replaces the baseline swirl settings in the *recommend* object. This rule recommends a maximum shroud bias test so the value of the shroud bias is set to -1. This notifies the ***main*** module to instruct the user to bias the shrouds toward the top of the furnace as much as possible. Control is then passed to the ***main*** module.

Once control is returned and the ***retrieve-data*** rule is fired, the swirl tests are again run for the current shroud settings. Once the swirl tests are completed the ***third-shroud-test*** rule is fired and replaces the baseline swirl settings in the *recommend* object. This rule modifies the *recommend* object with shroud settings equal to the average of the first two shroud settings. Next, control is passed to the ***main*** module to conduct the test. When control is returned, the same swirl tests are again run at the current shroud settings. This completes the testing for the values from the *optimum1* object.

Once testing is complete, the settings which produce the minimum NOx and most swirl are chosen as the best. If the goal is to minimize heat rate at a target NOx level, the ***curve-fit*** rule then determines the shroud bias which is needed to achieve the target NOx. The control settings are then placed in the *optimum1* or *optimum2* objects. If the previous series of tests were from the values stored in *optimum1*, the entire procedure repeats itself using the values from the *optimum2* object.

When all of the testing is done, the ***modify-average*** rule is fired. This rule modifies the *optimumave* object with the average of the *optimum1* and *optimum2* objects. It also modifies the ***par-eval*** slot to hold *optimized-par*. Control is then returned to the ***main*** module.

Overfire Air Damper Distribution Module

The ***overfire air damper distribution*** (***OFAD***) module has two initial facts in its knowledge base. The first fact indicates the minimum overfire air distribution possible during a test, and the second indicates the minimum overfire air increment.

When control is first passed to the ***OFAD*** module, the rule called ***prepare-for-testing*** is fired (refer to Figure 22 for flow chart). This rule modifies the ***par-eval*** slot of the *recommend* object to *OFAD* which indicates parametric tests of overfire air distribution are being performed. A file called *ofad.dat* is then created and a "1" is written to it. This file holds the test data for this module. Constraints read in by the ***overfire air / O2*** module are also used in this module. A fact is then asserted to begin the testing, and the ***first-test*** rule is fired.

The ***first-test*** rule copies the control settings tested so far into the *recommend* object. The ***OFAD*** module runs for values from the *optimum1* and then *optimum2* objects. Thus, the module is essentially run twice with different control settings each time. Next, the experiment number is incremented and its fact re-asserted. The OFA increment used is determined and the new overfire air damper openings are determined. This first test is a positive bias test, where more air is biased to the top of the furnace. The new

OFA-bias is then calculated and placed in the *recommend* object with the new settings. Control is then passed to the ***main*** module.

Once the test has been conducted and control returned, the ***retrieve-data*** rule is fired. This rule is always the first rule to be fired once control is returned to the module and no violations have occurred. This rule modifies the *current* object to be *previous* and updates the test number. The minimum NOx achieved this far, and the corresponding OFAD bias is asserted as a fact. Next, the OFAD bias and NOx from the previous test are written to the *ofad.dat* file. It then determines the next rule to be fired and asserts it as a fact.

Once ***retrieve-data*** has fired, the rule called ***second-test*** is fired. This rule is exactly the same as the ***first-test*** rule except, a negative OFA-bias is recommended. Control is then returned to the ***main*** module. Once the second test has been finished, the ***midpoint-test*** rule is fired. This rule averages the results from the first two tests and then recommends the midpoint overfire air openings and bias. The midpoint is skewed twenty percent from the baseline values to create a slightly positive OFA-bias. This is done to assure three distinct test points are collected. These settings are then placed in the *recommend* object and control is returned to the ***main*** module. A positive OFA-bias indicates more air flows toward the top of the furnace, while a negative OFA-bias indicates the opposite affect.

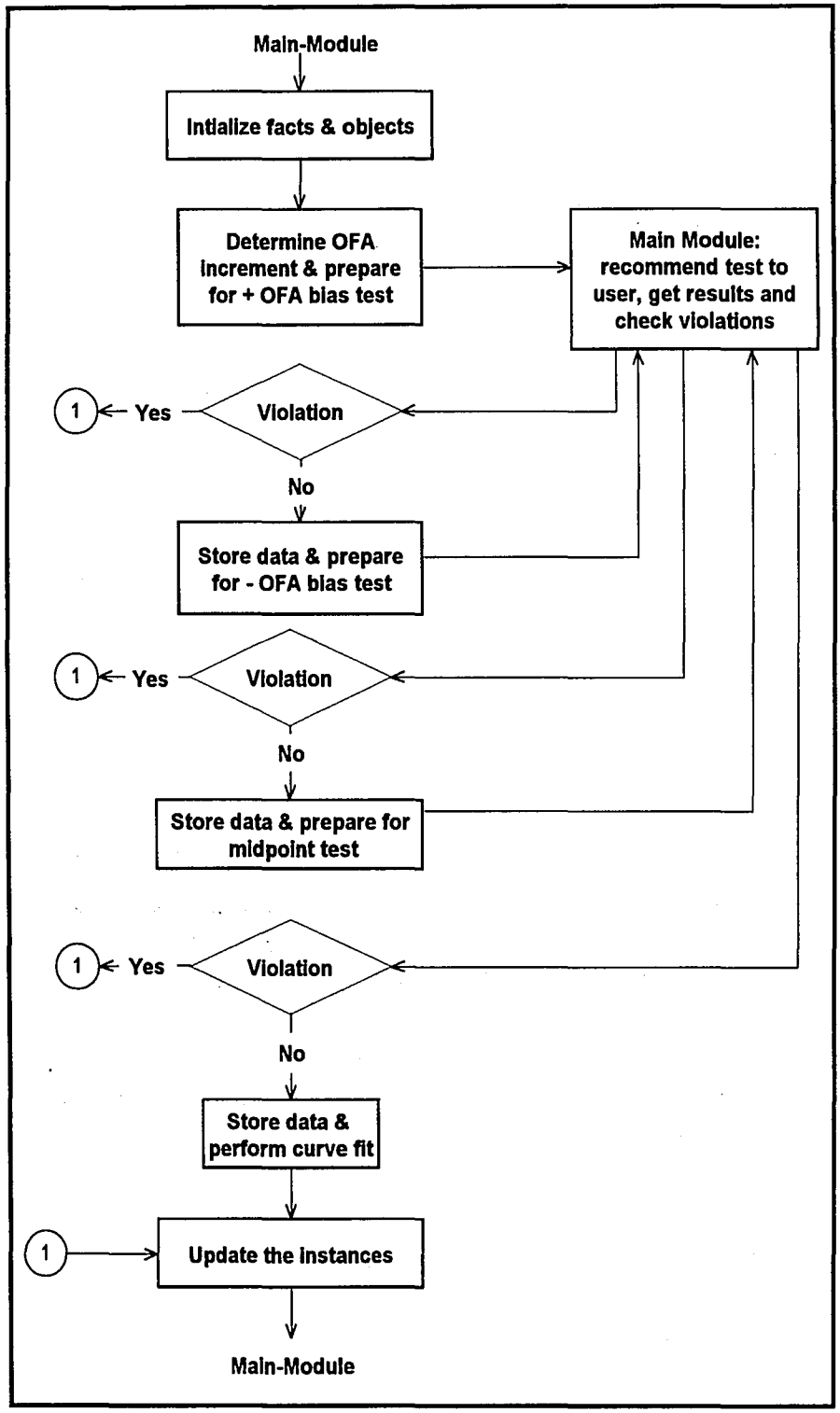

Figure 22 - Overfire air damper distribution module flow chart

If a limit violation occurs during any test, the ***end-module-violation*** rule is fired. This rule updates the *current* object to be *previous* and modifies the ***par-eval*** slot to hold *optimized-par.* The module is then finished and control is returned to the ***main*** module.

Once all three tests have successfully completed, the ***curve-fit*** rule is fired if the goal is to achieve a target NOx level. If the goal is to achieve minimum NOx, the ***modify-optimums*** rule is fired right after the third test is completed. This rule determines the approximates OFA-bias and the corresponding overfire air damper settings to achieve this NOx.

The ***modify-optimums*** rule is then fired. This rule simply places the best control settings into the *optimum1* or *optimum2* object. If the series of tests used starting control settings from the *optimum1* object, then the best OFA-bias and damper settings are stored in *optimum1*. If the series of tests used starting control settings from the *optimum2* object, then the best OFA-bias and damper settings are stored in *optimum2*. Once two series of tests are run, the rule called ***modify-optave*** is fired. This rule averages the OFAD control settings and places these in the *optimumave* object. The module is then finished and control is returned to the ***main*** module.

Mills Module

The ***mills*** module has an initial fact in its knowledge base indicating the total number of mills tests in a series. Another fact stating the minimum NOx and mill-bias is also contained in the ***mills*** module.

When control is first passed to the ***mills*** module, the ***prepare-for-tests*** rule is fired (refer to Figure 23 for flow chart). This rule first reads in any mill constraints from a file called *millcon.dat*. This information is then asserted as facts to be used throughout the module. Next, the ***par-eval*** slot is then modified to hold the value of *mills*. After that, a file called *mill.dat* is created and a "1" is written to it. This file will also hold all of the mill test data to be analyzed at the end. Finally, some facts are asserted for initialization purposes and to fire the next rule.

The rule called ***do-baseline-mills-test*** is then fired. This rule copies the control settings from the *optimum1* object into the *recommend* object. The mills settings are maintained at the baseline settings. Control is then passed to the ***main*** module to conduct the test.

Once the test has been conducted and control returned, the ***retrieve-data*** rule is fired. This rule is always the first rule to be fired once control is returned to the module and no violations have occurred. This rule modifies the *current* object to be *previous* and updates the test number. The previous test data are then copied into the *last-test* object. The minimum NOx achieved thus far and the corresponding mill- bias are asserted as a fact. Next, the mill-bias and NOx from the previous test are written to the *mill.dat* file. Then the rule determines the next rule to be fired and asserts a fact stating the rule to be fired.

Once the baseline mills test has been completed and the data retrieved, the first test in which mill biasing will be recommended is done. The rule which recommends this test is called ***first-test-in-series***. First the mill-bias is set to -1.0 and updated in the *recommend*

object. This notifies the ***main*** module that this is the first mills test so the user is notified to bias the mills as much as possible toward the bottom of the furnace. The fact counting the experiment number and series number is then updated and re-asserted. Control is then returned to the ***main*** module to conduct the test.

After control returns and the ***retrieve-data*** rule has fired and if no limit violations have occurred, a rule called ***begin-next-test*** is fired. This rule then determines two more test points in between the baseline settings and the first mills test settings. These two points are 2/3 the maximum mill bias and 1/3 the maximum mill bias. These test points are then asserted as facts. Next, the first recommended mill-bias along with its corresponding mill settings are placed in the *recommend* object. Experiment number and series number facts are then asserted and control is returned to the ***main*** module. When control returns and if no violations have occurred, this rule is fired a second time. This time the second recommended test point is placed in the *recommend* object and control is passed to the ***main*** module.

If a limit violation occurs, the rule called ***mill-violation*** is fired. This rule modifies the *current* object to be *previous* and updates the test number. The ***mills*** module is then finished and control is returned to the ***main*** module.

During the testing procedure, the *last-test* object is constantly compared to the *current* object to see if a significant NOx change is being achieved. If a significant NOx change is not achieved, a violation is asserted stating this and the ***mill-violation*** rule is fired to end the testing.

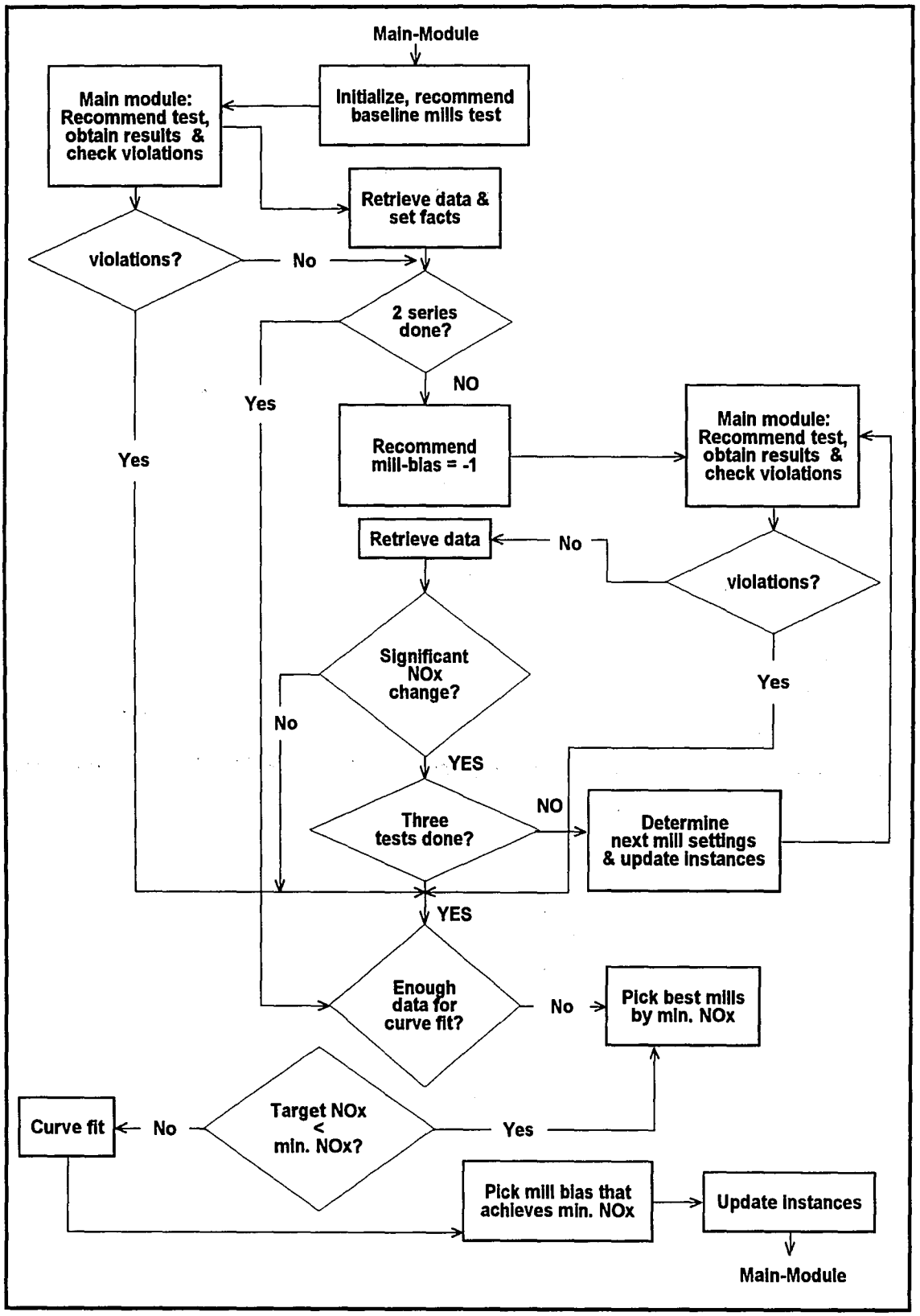

Figure 23 - Mills module flow chart

Once a total of three mills tests, not counting the baseline test, has been performed, the process repeats itself with control settings from the *optimum2* object. This is to build a test data base with enough data points for the mill-bias parameter. If only a total of four mills tests were conducted, the neural network / optimization routine may produce unreliable results. Once the entire process repeats itself, the testing is complete and the best control settings must be chosen.

If the goal is to minimize heat rate subject to a target NOx, the rule called ***curve-fit*** is fired, otherwise the ***modify-optimums*** rule is fired. The ***curve-fit*** rule determines the approximate mill-bias and corresponding mill settings to achieve this NOx. The ***modify-optimums*** rule places the best mill-bias and mill settings into the *optimum1*, *optimum2* and *optimumave* objects. It also modifies the ***par-eval*** slot to hold *optimized-par* to indicate the mill-bias parameter has been tested. The module is then finished and control returned to the ***main*** module. The next section discusses simulated results of the advisor.

Results

While the expert system testing advisor is not ready to be tested at a power plant, it still must be validated to assure there are no errors and the knowledge base recommends proper tests to reduce NOx. The most important step in the validation process is to validate the logic, which makes up the knowledge base, to determine whether it functions as intended. The approach used for validating the expert system was to run the advisor using actual test data collected at Chalk Point Unit 2. Using these data, linear correlations were generated which approximate the effect each parameter has on NOx. Liner correlations were used because they are easy to generate and provide enough insight to determine if the recommended test points will reduce NOx. Figures 24 through 27 show plots of the data collected at Chalk Point and the corresponding equations of the best fit lines for each parameter.

The advisor was run for four distinct trials. The first trial run assumed it was being used at Chalk Point Unit 2 and each parameter was tested. This trial run used a goal to minimize NOx. The second trial run was the same as trial run 1, however, the goal was to minimize heat rate subject to a target NOx of 0.85 lb/MBtu. Although this was difficult to test without actual heat rate data, approximate values were used by comparing similar control settings with the actual data. The third trial run was used to determine if the ***significant parameter*** module ran as expected and recommended tests which produced a NOx reduction. The fourth and final trial run tested the overfire air distribution module assuming three rows of overfire air ports existed. This trial was to determine if the

overfire air distribution module ran as intended and recommended tests which would reduce NOx.

Table 1 shows the actual baseline conditions which were used to initialize the advisor. Once the advisor was initialized, trial run 1 was begun. The entire testing sequence of trial run 1 is shown in Table 2. This table shows in detail each test which was run and the corresponding control settings for each test. The secondary air module was run using only the average optimum settings from the *optimumave* object rather than running two separate series of tests, one using the optimum settings from the *optimum1* object and one using the optimum settings from the *optimum2* object. This was done to save testing time and to show that if the module worked correctly for one series of tests it would work for two series of tests. If two series had been performed, the total number of tests would have exceeded fifty.

Figures 28 through Figure 32 show various plots for each parameter tested. Since trial runs 1 and 2 are very similar, the figures are the same. Each plot verifies that the logic in the knowledge base does recommend control settings which drive NOx downward while maintaining safe boiler operation. Thus, the knowledge base has been successfully validated by trial runs 1 and 2. Figure 28 shows NOx versus economizer-oxygen level. The tests stopped at an O_2 of 2.65% because a violation occurred. When this violation occurred, the user was properly notified and given the option of suspending the testing or continuing. Similar checks in every other module were made when violations occurred..

Trial run 2 was also successful in achieving a target NOx and minimizing heat rate. Due to the correlations and the artificial heat rates used, the results of trial runs 1 and 2

are essentially the same. During this trial run, the external program ***v1least.exe*** and the rules to use this program operated as intended and chose correct control settings.

The next step in the validation process was to assure the ***significant parameter*** module operated as expected and recommended test points which would reduce NOx. This trial run, like trial runs 1 and 2, was operated using the baseline conditions shown in Table 1. Table 3 shows the test sequence recommended by the module. This table shows that the module did operate as it was intended and successfully recommended extreme parameter tests to produce a significant NOx change. This table shows that as each parameter was tested, the remaining control settings were maintained at baseline conditions, this was done to isolate the effect each parameter had on NOx. Figure 33 shows NOx versus test number for these tests. This figure shows decreasing economizer-oxygen decreases NOx, opening the overfire air dampers decreases NOx, biasing the secondary air towards the top of the furnace decreases NOx and biasing the coal flow towards the bottom of the furnace decreases NOx. This trends validate the logic which was used to create the knowledge base of this module.

The final step in the validation process was to determine if the overfire air distribution module operated correctly and recommended test points which lowered NOx. In order for this module to run, the advisor was configured to have three rows of overfire air ports rather than one row as in Chalk Point Unit 2. The same baseline conditions shown in Table 1 were again used.. Table 4 shows the test sequence of the overfire air distribution module. This table verifies that two series of tests were performed, one series using control settings from the *optimum1* object and one series using control settings from

the *optimum2* object. This table also indicates a positive bias test, negative bias test, and a midpoint bias test was performed for each series as was intended. Finally, it shows that only the overfire air dampers and the bias is modified in this module. Figure 34 shows the overfire air bias parameter and corresponding NOx versus test number. This figure shows a positive bias decreases NOx and a negative bias increases NOx. This indicates that the knowledge used to create this module is correct and the module works as intended.

Baseline Conditions
Load, 355 MW
Economizer-oxygen, 4%
Overfire air dampers, 100%
Shrouds, 100%
Swirl Vanes, 30 deg.
No mill bias
Baseline NOx = 0.93 lb/MBtu
Target NOx = 0.85 lb/MBtu

Table 1 - Baseline operating conditions

Test Sequence - Trial 1 & Trial 2								
#	Module	O2	Avg. OFA	shrouds	swirls	shroud bias	mills	mill bias
1	baseline	4	100	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
2	O2	4	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
3	O2	3.5	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
4	O2	3.25	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
5	O2	3.05	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
6	O2	2.85	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
7	O2	2.65	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
8	OFA/O2	3.05	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
9	OFA/O2	3.05	45	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
10	OFA/O2	3.25	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
11	OFA/O2	3.25	55	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
12	OFA/O2	3.25	60	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
13	OFA/O2	3.25	65	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
14	OFA/O2	3.25	70	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
15	OFA/O2	3.25	75	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
16	OFA/O2	3.45	70	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
17	OFA/O2	3.45	75	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
18	OFA/O2	3.45	80	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
19	OFA/O2	3.65	80	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
20	OFA/O2	3.65	85	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
21	SEC-AIR	2.85	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
22	SEC-AIR	2.85	50	100,100,100,100,100,000	25	0	40,40,40,40,40,40	0
23	SEC-AIR	2.85	50	100,100,100,100,100,000	37	0	40,40,40,40,40,40	0
24	SEC-AIR	2.85	50	100,100,100,100,100,000	40	0	40,40,40,40,40,40	0
25	SEC-AIR	2.85	50	100,80,60,60,80,100	30	0.4	40,40,40,40,40,40	0
26	SEC-AIR	2.85	50	100,80,60,60,80,100	25	0.4	40,40,40,40,40,40	0
27	SEC-AIR	2.85	50	100,80,60,60,80,100	37	0.4	40,40,40,40,40,40	0
28	SEC-AIR	2.85	50	100,80,60,60,80,100	40	0.4	40,40,40,40,40,40	0
29	SEC-AIR	2.85	50	100,90,80,80,90,100	30	0.2	40,40,40,40,40,40	0
30	SEC-AIR	2.85	50	100,90,80,80,90,100	25	0.2	40,40,40,40,40,40	0
31	SEC-AIR	2.85	50	100,90,80,80,90,100	37	0.2	40,40,40,40,40,40	0
32	SEC-AIR	2.85	50	100,90,80,80,90,100	40	0.2	40,40,40,40,40,40	0
33	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	40,40,40,40,40,40	0
34	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	32,44,44,32,44,44	-0.1
35	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	35,42,42,35,42,42	-0.6
36	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	38,41,41,38,41,41	-0.03
37	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	32,44,44,32,44,44	-0.1
38	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	35,42,42,35,42,42	-0.6
39	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	38,41,41,38,41,41	-0.03

Table 2 - Test sequence of advisor

Baseline Conditions
Load, 355 MW
Economizer-oxygen, 4%
Overfire air dampers, 100%
Shrouds, 100%
Swirl Vanes, 30 deg.
No mill bias
Baseline NOx = 0.93 lb/MBtu
Target NOx = 0.85 lb/MBtu

Table 1 - Baseline operating conditions

Test Sequence - Trial 1 & Trial 2								
#	Module	O2	Avg. OFA	shrouds	swirls	shroud bias	mills	mill bias
1	baseline	4	100	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
2	O2	4	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
3	O2	3.5	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
4	O2	3.25	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
5	O2	3.05	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
6	O2	2.85	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
7	O2	2.65	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
8	OFA/O2	3.05	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
9	OFA/O2	3.05	45	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
10	OFA/O2	3.25	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
11	OFA/O2	3.25	55	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
12	OFA/O2	3.25	60	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
13	OFA/O2	3.25	65	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
14	OFA/O2	3.25	70	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
15	OFA/O2	3.25	75	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
16	OFA/O2	3.45	70	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
17	OFA/O2	3.45	75	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
18	OFA/O2	3.45	80	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
19	OFA/O2	3.65	80	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
20	OFA/O2	3.65	85	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
21	SEC-AIR	2.85	50	100,100,100,100,100,000	30	0	40,40,40,40,40,40	0
22	SEC-AIR	2.85	50	100,100,100,100,100,000	25	0	40,40,40,40,40,40	0
23	SEC-AIR	2.85	50	100,100,100,100,100,000	37	0	40,40,40,40,40,40	0
24	SEC-AIR	2.85	50	100,100,100,100,100,000	40	0	40,40,40,40,40,40	0
25	SEC-AIR	2.85	50	100,80,60,60,80,100	30	0.4	40,40,40,40,40,40	0
26	SEC-AIR	2.85	50	100,80,60,60,80,100	25	0.4	40,40,40,40,40,40	0
27	SEC-AIR	2.85	50	100,80,60,60,80,100	37	0.4	40,40,40,40,40,40	0
28	SEC-AIR	2.85	50	100,80,60,60,80,100	40	0.4	40,40,40,40,40,40	0
29	SEC-AIR	2.85	50	100,90,80,80,90,100	30	0.2	40,40,40,40,40,40	0
30	SEC-AIR	2.85	50	100,90,80,80,90,100	25	0.2	40,40,40,40,40,40	0
31	SEC-AIR	2.85	50	100,90,80,80,90,100	37	0.2	40,40,40,40,40,40	0
32	SEC-AIR	2.85	50	100,90,80,80,90,100	40	0.2	40,40,40,40,40,40	0
33	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	40,40,40,40,40,40	0
34	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	32,44,44,32,44,44	-0.1
35	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	35,42,42,35,42,42	-0.6
36	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	38,41,41,38,41,41	-0.03
37	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	32,44,44,32,44,44	-0.1
38	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	35,42,42,35,42,42	-0.6
39	MILLS	2.95	50	100,80,60,60,80,100	40	0.4	38,41,41,38,41,41	-0.03

Table 2 - Test sequence of advisor

Significant Parameter Module Tests - Trial 3			
Parameter	**Setting**	**Other parameters**	**NOx**
baseline	baseline	baseline	0.93
O2	2.85%	baseline	0.841
OFA	100%	baseline	0.921
OFA	40%	baseline	0.943
Shroud bias	0.4	baseline	0.875
Mill bias	-0.1	baseline	0.861

Table 3 - Significant Parameter module sequence

OFAD Tests - Trial 4									
#	**Module**	**O2**	**Avg. OFA**	**OFA-Bias**	**shrouds**	**swirls**	**shroud bias**	**mills**	**mill bias**
15	OFAD	2.85	50	25	100	30	0	40,40,40,40,40,40	0
16	OFAD	2.85	50	-25	100	30	0	40,40,40,40,40,40	0
17	OFAD	2.85	50	10	100	30	0	40,40,40,40,40,40	0
18	OFAD	3.05	50	25	100	30	0	40,40,40,40,40,40	0
19	OFAD	3.05	50	-25	100	30	0	40,40,40,40,40,40	0
20	OFAD	3.05	50	10	100	30	0	40,40,40,40,40,40	0

Table 4 - Overfire air distribution module sequence

Significant Parameter Module Tests - Trial 3			
Parameter	Setting	Other parameters	NOx
baseline	baseline	baseline	0.93
O2	2.85%	baseline	0.841
OFA	100%	baseline	0.921
OFA	40%	baseline	0.943
Shroud bias	0.4	baseline	0.875
Mill bias	-0.1	baseline	0.861

Table 3 - Significant Parameter module sequence

OFAD Tests - Trial 4									
#	Module	O2	Avg. OFA	OFA-Bias	shrouds	swirls	shroud bias	mills	mill bias
15	OFAD	2.85	50	25	100	30	0	40,40,40,40,40,40	0
16	OFAD	2.85	50	-25	100	30	0	40,40,40,40,40,40	0
17	OFAD	2.85	50	10	100	30	0	40,40,40,40,40,40	0
18	OFAD	3.05	50	25	100	30	0	40,40,40,40,40,40	0
19	OFAD	3.05	50	-25	100	30	0	40,40,40,40,40,40	0
20	OFAD	3.05	50	10	100	30	0	40,40,40,40,40,40	0

Table 4 - Overfire air distribution module sequence

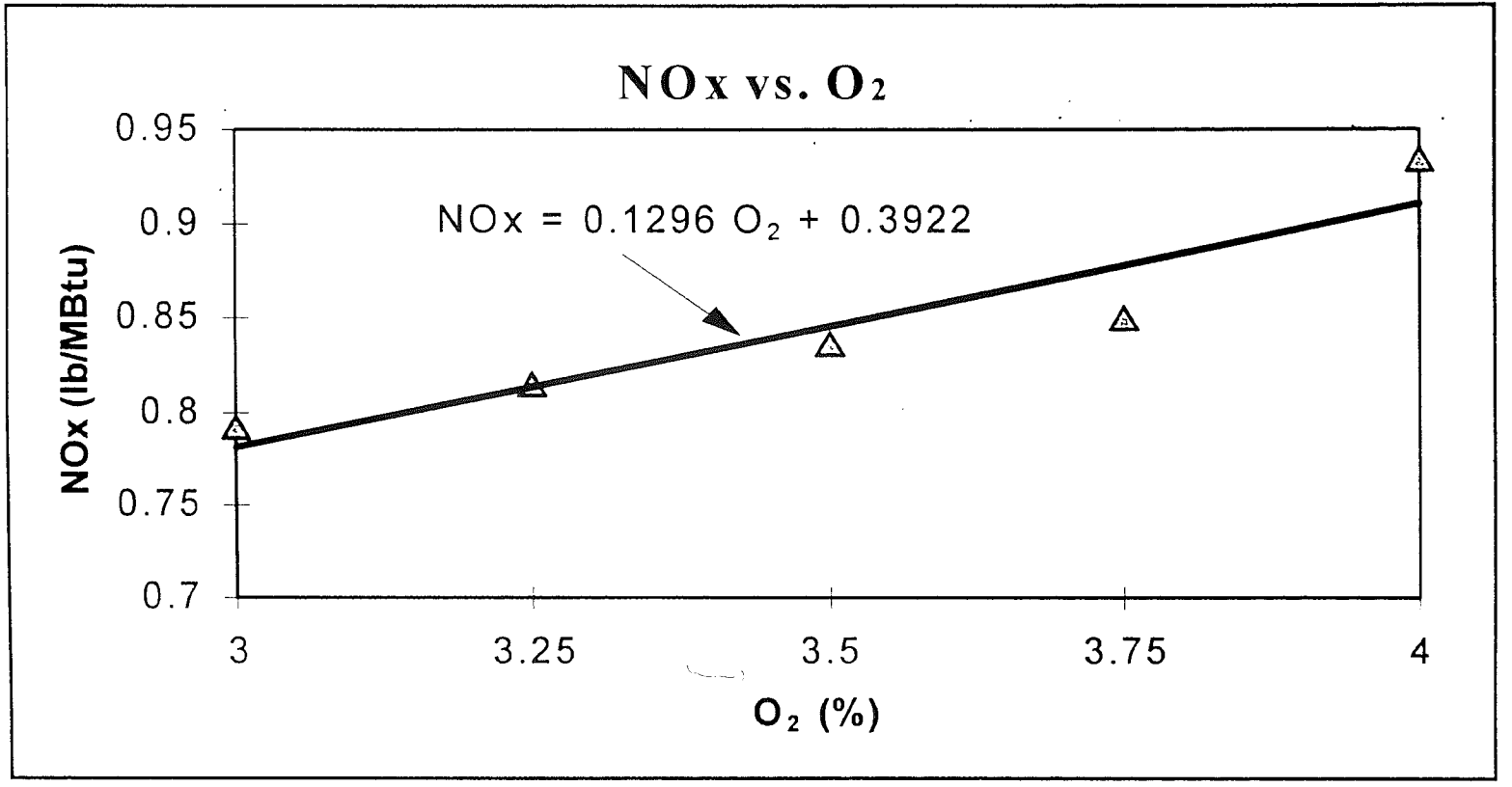

Figure 24 - NOx vs. O_2 (linear curve fit of power plant test data)

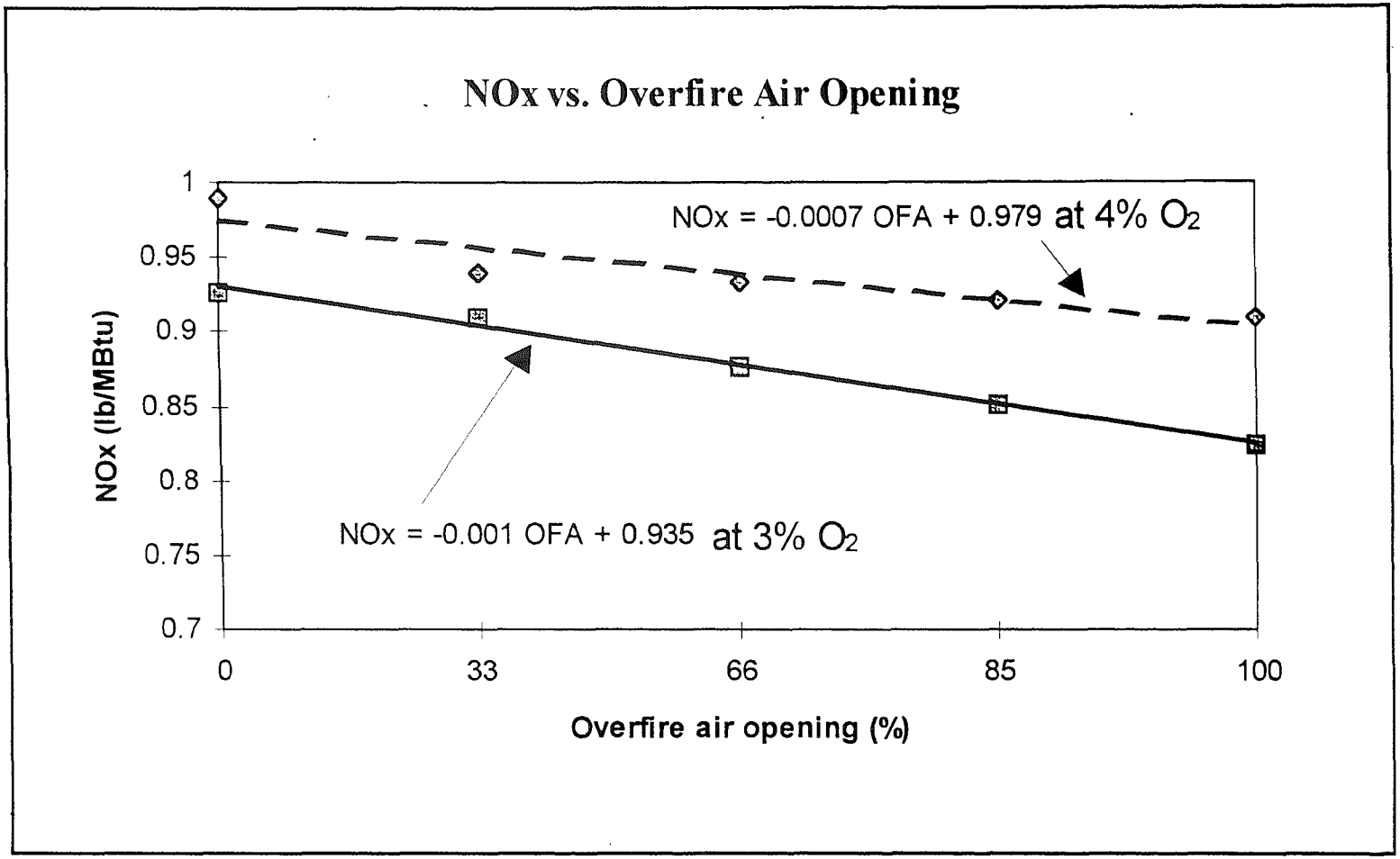

Figure 25 - NOx vs. Overfire air (linear curve fit of power plant test data)

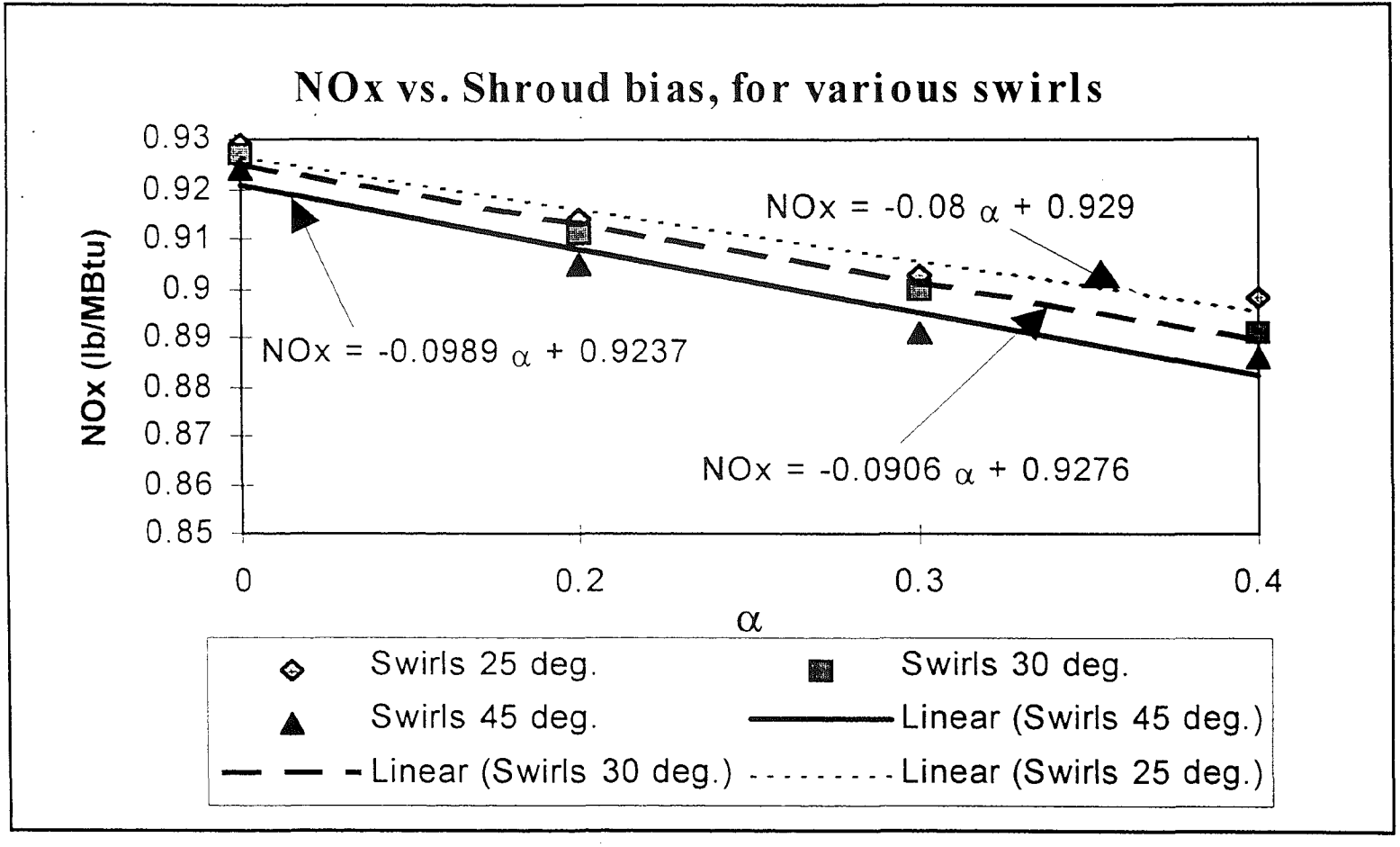

Figure 26 - NOx vs. Shroud bias (linear curve fit of power plant test data)

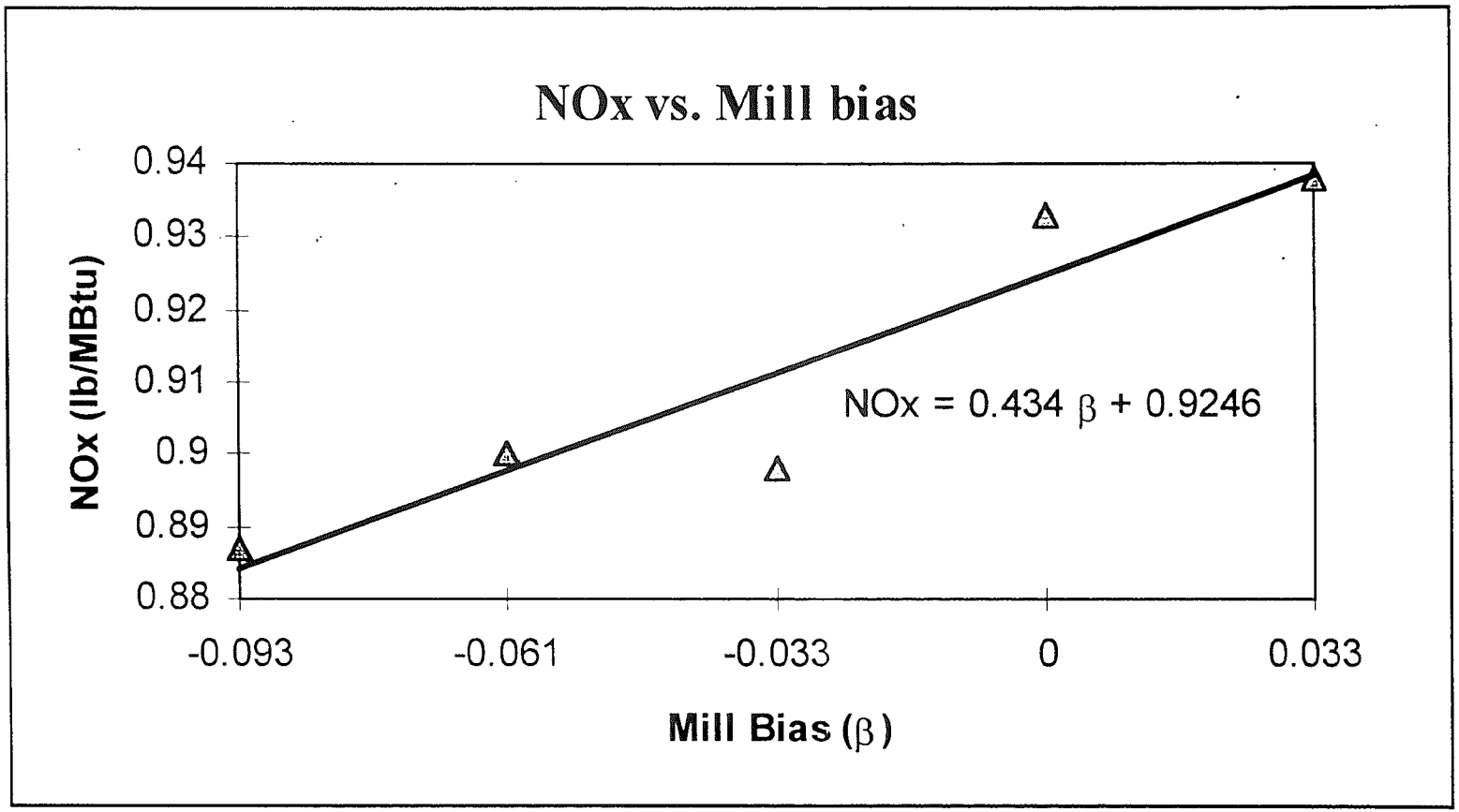

Figure 27 - NOx vs. Mill bias (linear curve fit of power plant test data)

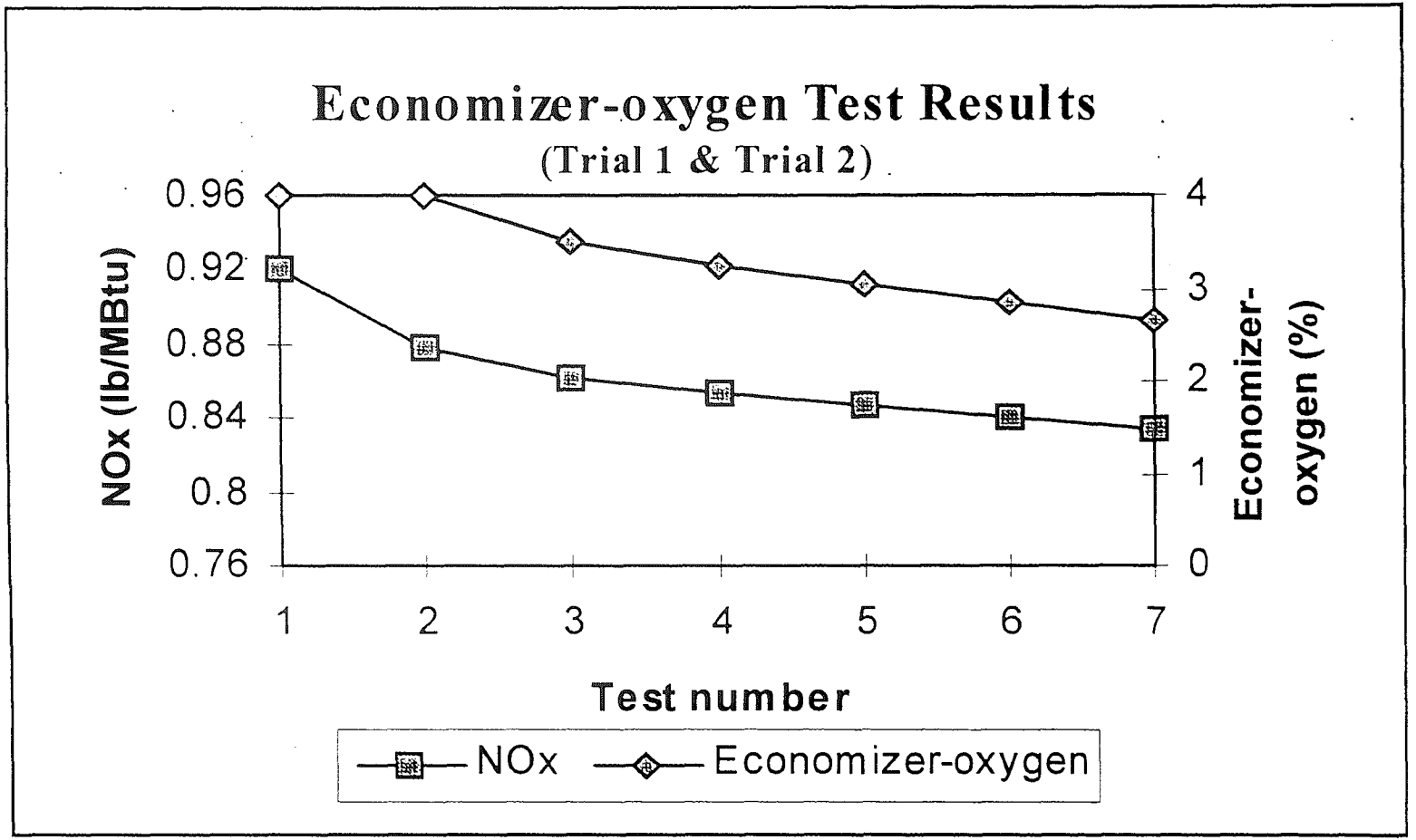

Figure 28 - Economizer-oxygen test results (Trial 1 & 2)

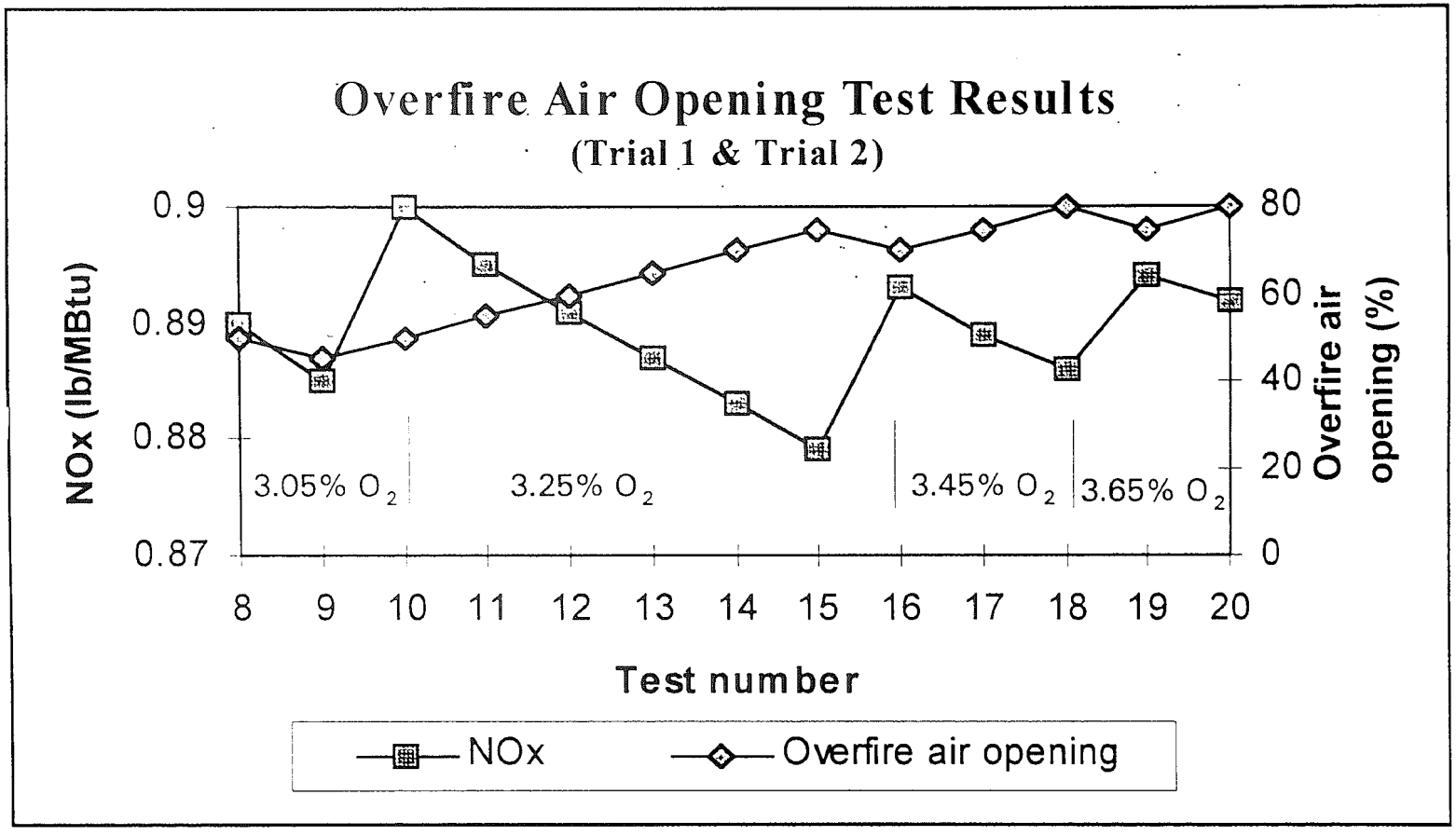

Figure 29 - Overfire air test results (Trial 1 & 2)

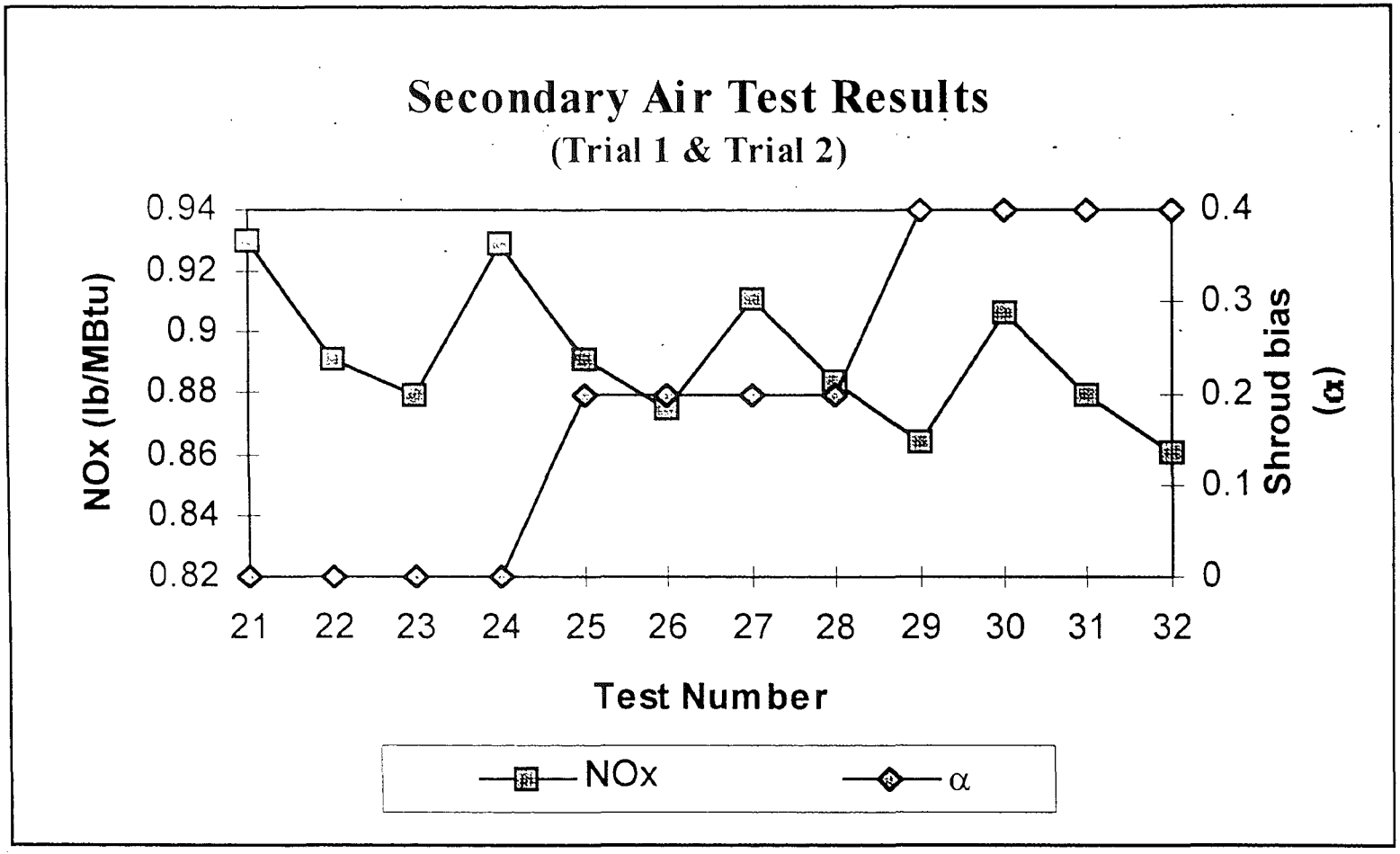

Figure 30 - Secondary air test results (Trial 1 & 2)

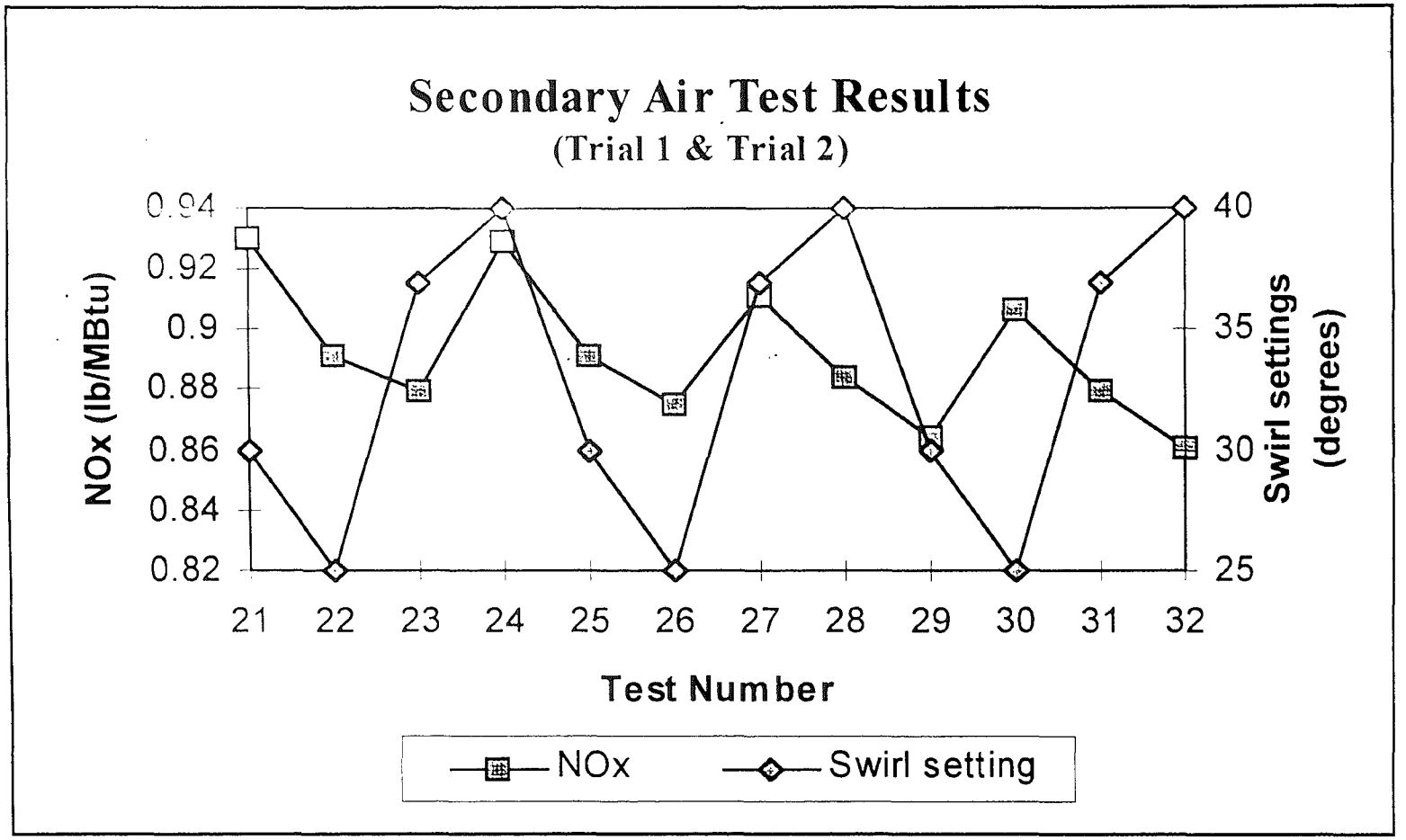

Figure 31 - Secondary air test results (Trial 1 & 2)

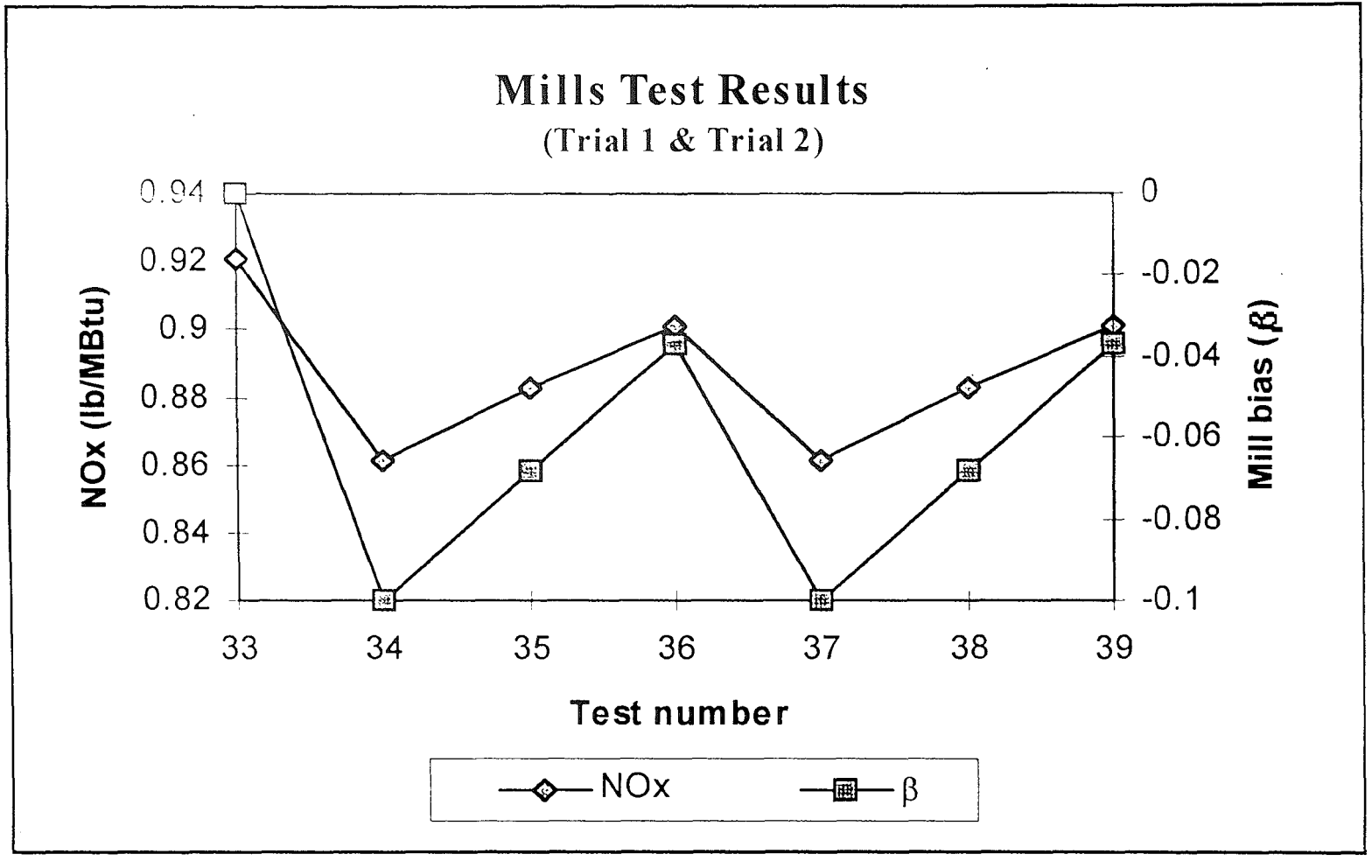

Figure 32 - Mills test results (Trial 1 & 2)

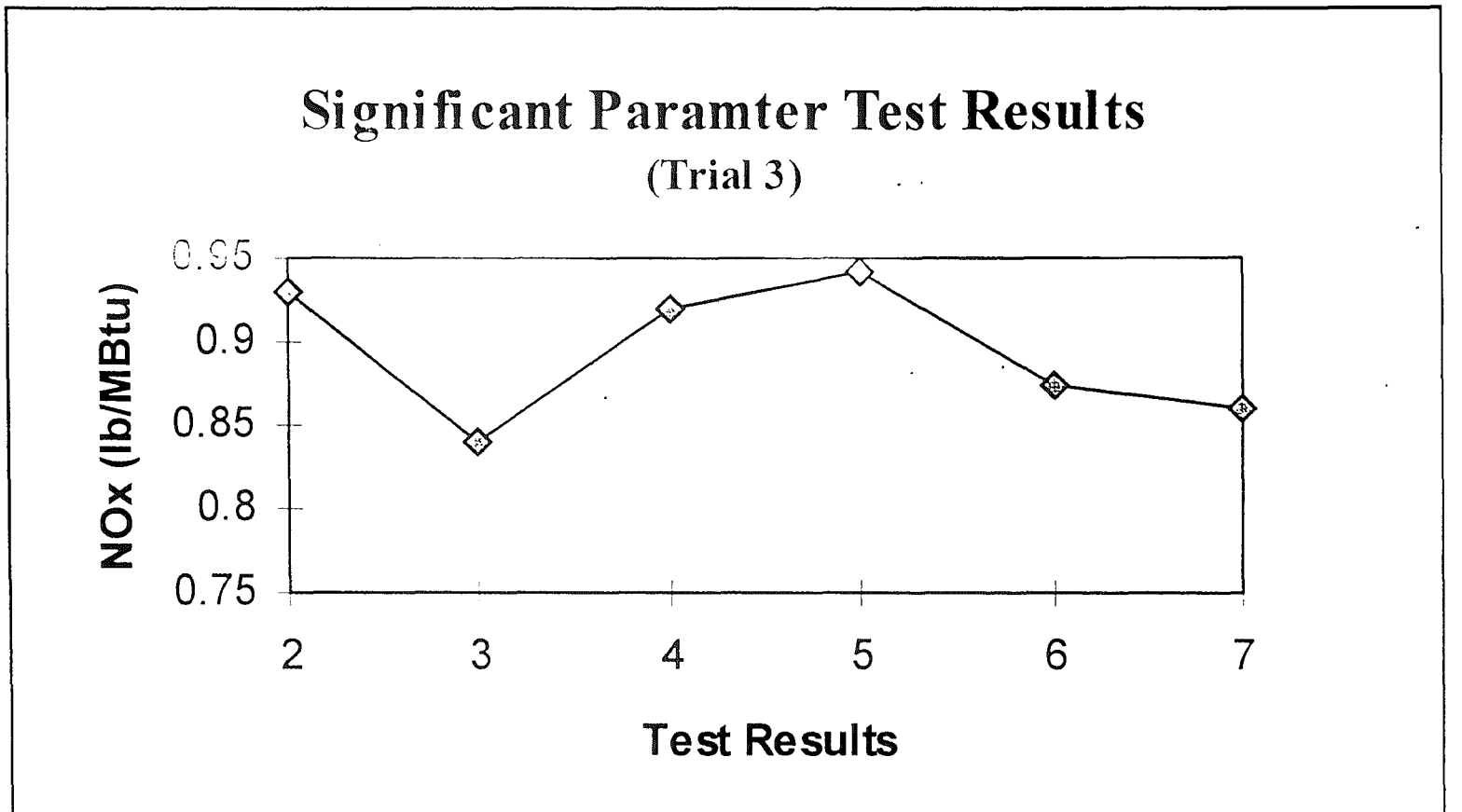

Figure 33 - Significant parameter test results (Trial 3)

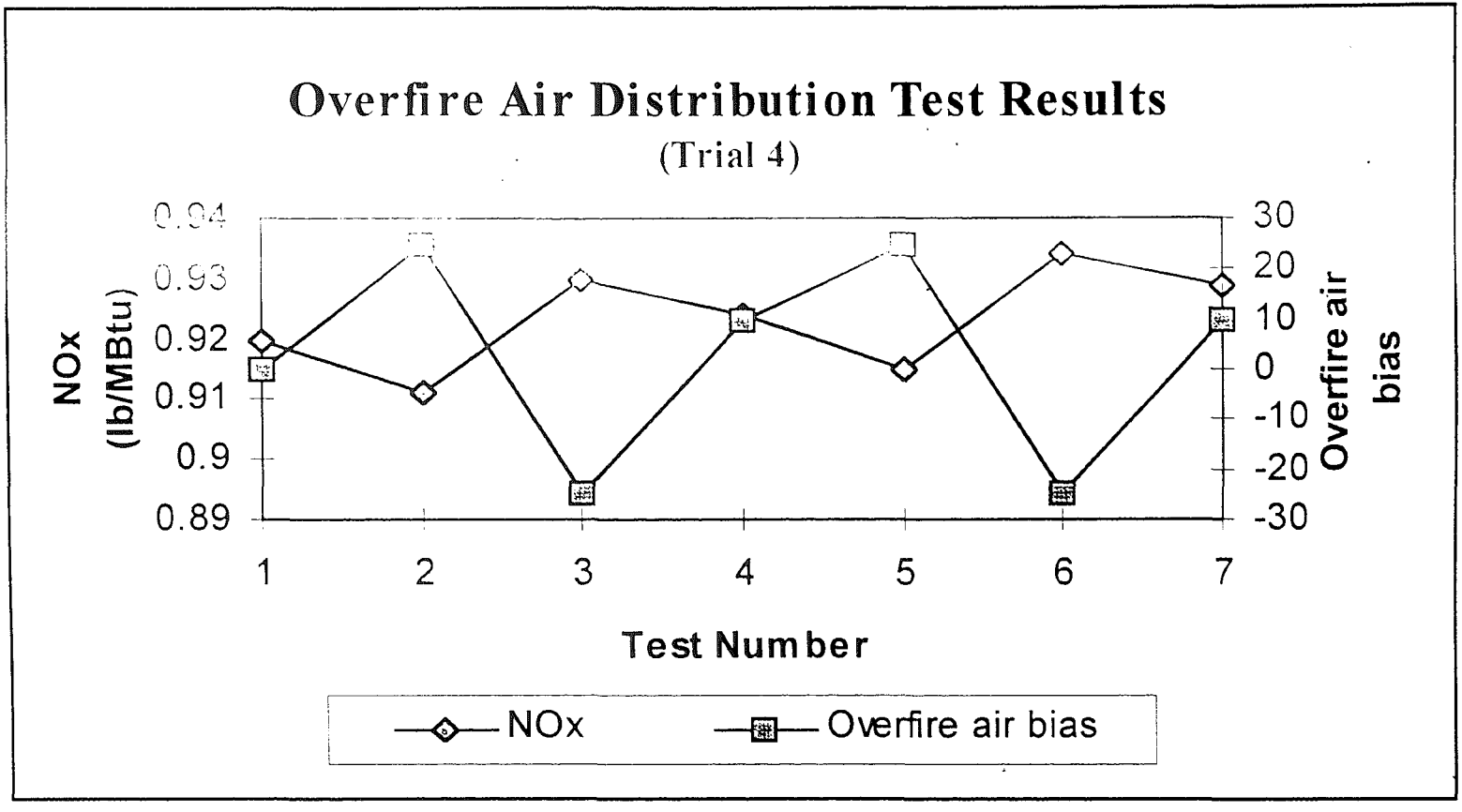

Figure 34 - Overfire air distribution test results (Trial 4)

OPTIMIZATION PRE-CONSTRAINER

Introduction

Once the wall-fired expert system finishes conducting the parametric tests on the boiler, the neural network/optimization code is run using the database from the tests. The neural network/optimization code, developed at the Energy Research Center, performs two tasks using the test data [22]. First, the neural network portion of the package uses the test database to learn or determine the complex relationships between the inputs and the outputs. This process is called training the neural networks. Once the networks are trained and created, the optimization code runs, using the networks just created, and determines the optimal control settings which meet the desired goal. A diagram of the entire process is shown in Figure 35.

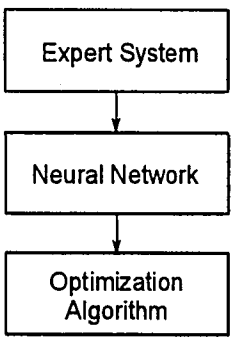

Figure 35 - Overall flow of optimization

The optimization portion of the software is a global optimum search algorithm which tries to find a global minimum solution over the parameter ranges. The disadvantage of this optimization routine is that it is a mathematical algorithm which finds

a mathematically sound solution, but one which may not be physically realistic. Although the boiler tests are typically conducted within safe limits, some of the test data are at the edge of the operating envelope. It is possible to get a combination of test parameters which provide unrealistic results, causing problems such as high LOI or high opacity.

There are possible solutions for avoiding this problem of the optimization algorithm choosing control settings on the outer edge of the tested range. One possible method is to conduct more conservative tests which always test in an area that is likely to be a good operating point. This approach undermines the purpose of the parametric testing strategy and may miss good operating points because it is too conservative. A second method utilizes the existing testing strategy, but before the test database is sent to the optimization algorithm, parameter ranges over which to search for a solution are recommended. Using this method, the range for each individual parameter can be chosen to always give valid operating points while preserving the larger database for use if less conservative results are desired.

In this thesis, the second method was chosen, developed and tested. Figure 36 shows the new format of the entire optimization process including this pre-constraining method. The next section discusses the design and development of the pre-constrainer.

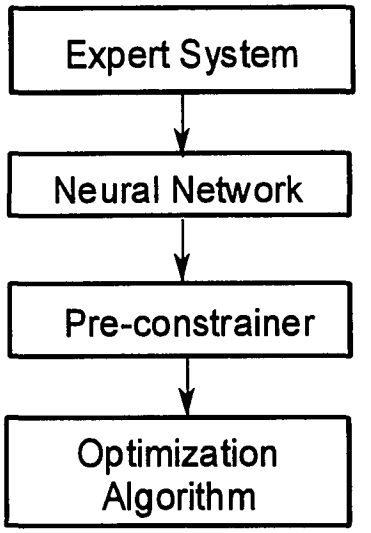

Figure 36 - Overall flow of optimization with pre-constrainer

Description of logic

When the optimization software is run without the pre-constrainer, it looks for a solution over the entire test range for each parameter. This can result in solutions which are beyond the range at which tests were conducted. In cases where there is an insufficient number of data, attempts to obtain optimal solutions can result in unreliable solutions.

After performing optimizations on test data from three different boilers over wide ranges of target NOx levels and test parameters, a method was developed for determining parameter ranges which proved to be realistic safe operating conditions. The method developed, which is dependent on the target NOx level and the number of parameters tested, is based on statistical analysis and knowledge generated from test results.

The first step of the development process was to determine how many data are sufficient to consistently obtain reliable solutions. The minimum number of test points required to obtain reliable solutions proved to be directly proportional to the number of parameters tested. The minimum number of points required was found to be equal to half the number of tested parameters. This result was determined by performing optimizations where parameter ranges were generated using different minimum required number of data points. First, one data point was considered the minimum; in the second case, the minimum required number of data points was equal to half the number of tested parameters; and the last case, the minimum required number of data points was equal to the number of tested parameters. When only one data point was required, it was very

difficult for the optimization algorithm to converge on a solution. When the minimum number required was equal to half the number of tested parameters, the optimization algorithm consistently found a solution over a wide range of NOx levels. When the number required was equal to the number of tested parameters, the results were similar to when only half was required, but solutions could not be found at the lower and upper NOx levels, where fewer test points are available. Thus, the minimum number of data points required was chosen as being equal to half the tested parameters. For example, if eight parameters were tested, there must be at least four test points at the target NOx level to obtain reliable results.

This approach has one limitation. Test points may not have NOx levels exactly equal to the target NOx level, causing the number of test points to be zero, so no optimization can be performed. To allow for some variation between tested NOx levels and target NOx levels, a tolerance, ΔNOx, applied to the target NOx level is used. A tolerance of 0.01 lb/MBtu is used because it is a small variation in NOx, but still a significant variation. Thus, a total NOx range of 0.02 lb/MBtu or $2 \cdot \Delta$NOx is used as the area within which to determine if the minimum number of data points in the range is present. Figure 37 shows a diagram of the target NOx with the tolerance applied. This method of using a tolerance was just as effective as without a tolerance, but allows for slight variations in NOx levels.

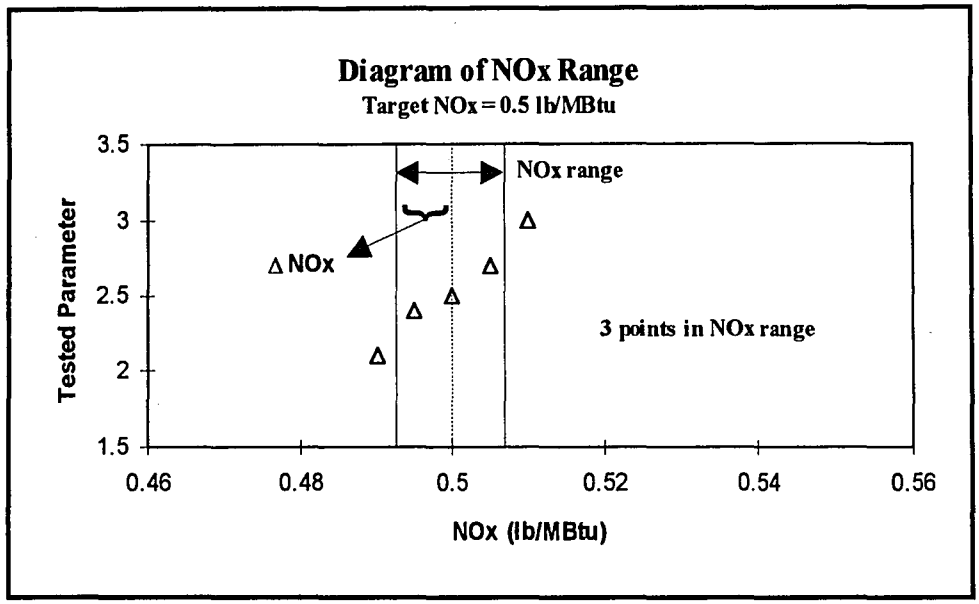

Figure 37 - Diagram of target NOx with tolerance applied

The last step in the development of the pre-constrainer was to determine how to choose ranges for each independent parameter. The range for each parameter must be conservative enough to eliminate unrealistic results, but not so conservative that it eliminates good operating points. The method chosen takes a statistical approach, combined with knowledge from the test data to choose the parameter ranges.

The approach is to calculate the average value and standard deviation for each tested parameter lying in the NOx range. Figure 38 shows the NOx range with test points in the range. Each test point has a value for O_2, mill bias, load ...,etc. Thus, the average value and standard deviation are calculated for each of the tested parameters. The range of each parameter is then determined by adding and subtracting some multiple of its standard deviation to its average. The two equations below show how the maximum and minimum value for each parameter are determined.

$$parameter_{i,\max} = average_i + K \cdot \sigma_i$$

$$parameter_{i,\min} = average_i - K \cdot \sigma_i$$

The constant K was varied from 1.0 to 3.0 in increments of 0.5, and optimizations were run to determine what value consistently gave the most realistic results. The value of K, which gave the most consistent and realistic results, was found to be 2.0.

Figure 38 shows a plot of O_2 versus NOx from a test set. This graph shows a diagram of the NOx range, the average value of O_2 for the points in the NOx range and the cutoff standard deviations where the minimum and maximum O_2 are found.

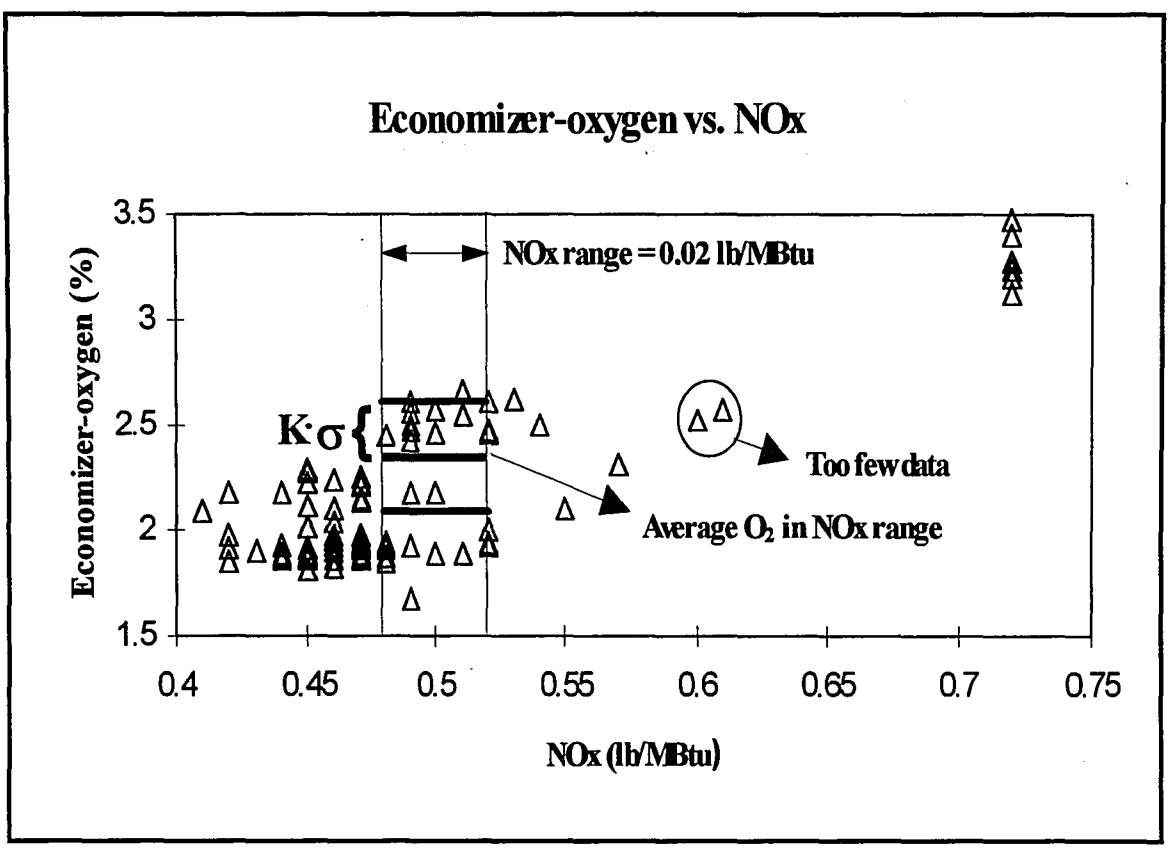

Figure 38 - Region from which the parameter ranges are determined

In this figure the target NOx is 0.5 lb/MBtu and the ΔNOx is 0.01 lb/MBtu on each side of the target NOx. Thus, the entire NOx band is 0.02 lb/MBtu wide. The cutoff standard deviation is $K \cdot \sigma$, where K equals 2.0. Therfore, the entire O_2 range is four times its standard deviation. The next section describes the operation of the pre-constrainer.

Software Description

The optimization pre-constrainer is a DOS based application written in ANSI C. The pre-constrainer does not interact with the user, but instead operates on files which the optimization code uses to determine the optimal control settings. The pre-constrainer reads three files for input, these are *coltest.ext*, *output1.txt* and *stdev.ext*, where *ext* represents any valid DOS file extension. This file extension, which is generated by the testing advisor, is read in from *output1.txt* file. The *coltest.ext* file is a text file consisting of one line of data. The line contains fourteen numbers each separated by a space. Valid numbers are 0, 1 and 2. A "zero," indicates that the corresponding column in *output1.txt* is an input to the neural networks, but was not tested by the expert system. A "one," indicates that the corresponding column in *output1.txt* is an input to the neural networks and was tested by the expert system. A "two," indicates that the corresponding column in *output1.txt* is an output from the neural networks.

The file *stdev.ext* is a text file consisting of one number. This number is the constant K defined in the previous section. Although it was found that the best value of K

is 2.0, other options are made available for instances when other values are desired. The final file read in by the pre-constrainer is the *output1.txt* file. This file is a text file containing information on the boiler type, file extension, target NOx and the test data itself.

The pre-constrainer also uses the file *data.txt* as the output file containing the constrained ranges and other values not used by the pre-constrainer, but used by the optimization routine. Figure 39 below shows the input and output of the pre-constrainer.

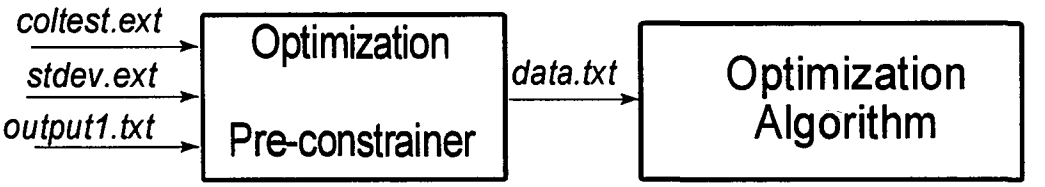

Figure 39 - Input and output of pre-constrainer

When the pre-constrainer is first run, it performs several initialization steps. First the file extension *.ext* from the *output1.txt* file is read in and stored. This extension is used to open the *coltest.ext* and *stdev.ext* files. Next, the constant K is read in from the *stdev.ext* file and stored. The pre-constrainer then reads the *coltest.ext* file and stores this information in an array. This information is used to determine the minimum number of data points in the NOx range required to proceed with the pre-constraining. Finally, the number of tests in the test file and the target NOx level are read in and stored. The program then allocates enough space in memory to hold the entire test set. If enough memory is not available, an error message is directed to the user and the program halts.

Next, the program uses the target NOx level and determines the NOx range by adding and subtracting the tolerance NOx of 0.01 lb/MBtu from the target NOx level. Once the NOx range is known, the number of test points which have NOx values which lie inside this NOx range are counted. If the number of points inside the range is less than the minimum required, a warning message is printed and the program halts.

If enough data points are in the range, space is allocated in memory to store just these points and the memory used to hold the entire test set is freed. Next, the average value and standard deviation for each parameter of the test points are determined. Once this information is known, the parameter ranges are calculated using the multiplication factor K, and the results are written to the *data.txt* file.

Before the program finishes executing, the memory still allocated is freed, and a file *grngdone.txt* is created with the word FINISHED in it. This file is used to indicate successful completion of the application.

Results

While developing the pre-constrainer, analyses were performed on several boiler test sets over a wide range of target NOx levels. Once the pre-constrainer was completed, it was tested on the same data for the same conditions to assure it was working properly and producing desirable parameter ranges. The results produced from the pre-constrainer were compared to results when no pre-constraining was used, and also to results when manual pre-constraining was used. No pre-constraining means the optimization code used the entire test range for each parameter to find a solution. Manual pre-constraining indicates pre-constraining that was done by choosing the range for each parameter by hand. The manually pre-constrained ranges were chosen so that the solutions were always within the tested range for each independent parameter. The manually pre-constrained ranges were then used by the optimization algorithm to generate the optimum control settings. These optimum control settings were then tested on the boiler to verify the boiler operated safely and with low NOx emissions. For this reason, manually pre-constrained ranges are considered the reference for comparison with the automatically pre-constrained ranges. The manually pre-constrained ranges were determined by research engineers at the Energy Research Center, based on years of boiler testing experience.

Test data from three different boilers were used to validate the pre-constrainer. Data from the Morgantown Power Plant units F1 and F2 were used, as well as data from the Potomac River Power Plant unit C5. The Morgantown data were obtained at a high

load of 585 MW and a low load of 475 MW. The Potomac River C5 data were obtained at 108 MW.

The results which will be discussed below were generated by using the pre-constrainer and optimization code over a wide range of target NOx levels. Each target NOx level produced a predicted optimal heat rate. These predicted heat rates and NOx levels were then plotted against the actual test data to verify the predicted NOx versus heat rates follow the actual. Generally, the predicted NOx versus heat rate curves follow the lower edge of the test data because the optimization algorithm is intended to find minimum heat rate values.

Figure 40 shows results from the Potomac River C5 unit. When no pre-constraining was performed, the predicted NOx versus heat rate curve closely follows the test data up to a target NOx of 0.50 lb/MBtu. Once the target NOx level is lower than 0.50 lb/MBtu the curve no longer follows the test data. In contrast to no pre-constraining, the curves for manually pre-constraining and automatically pre-constraining with the software follow the test data closely for all target NOx levels. Using a constant, K, of 1.5 proved to be nearly the same as manually pre-constraining. However, at NOx levels lower than 0.53 lb/MBtu, it was difficult to always converge on a solution. When a constant of 2.0 was used, the predicted results still followed the test data closely and it was considerably easier to get solutions at higher NOx levels.

Figure 41 is a continuation of the results for the C5 unit. This graph shows that as K increases, the results using the pre-constrainer begin to approach those when no pre-constraining was used. This is expected because as K increases, the parameter ranges

become larger. Thus, at some value of K, the pre-constrained ranges will be the same as the unconstrained ranges.

Figure 42 shows test results for the Morgantown F1 unit at loads of 475 MW and 585 MW. This figure is very similar to the previous ones for the Potomac River results. Again, the manual and automatically constrained results closely follow the test data, whereas the unconstrained results diverge from the test data at low NOx levels. The results for a constant of 2.0 are nearly identical to those of manually pre-constraining, while the results for K equal to 3.0 are closer to the unconstrained results.

Figure 43 shows test results for the Morgantown F2 unit. Again, this figure is very similar to the previous three figures. Results for manually pre-constraining and automatic pre-constraining, using a constant K equal to 2.0, are nearly identical and follow the actual test data remarkably well. When K equals 3.0 is used, the results begin to follow the unconstrained results at low NOx levels.

Although these results show constraining the parameter ranges produces results which closely follow results when manually pre-constraining was used, it was not known how close the solutions really are. To determine how good the pre-constrained results are, plots showing the percent error versus NOx for different values of K were created. Figure 44, for C5 data, shows for all values of K, the error of the solution is very small relative to heat rates in the 9000 kWh range. The figure also shows constraining the ranges reduced the error by more than 50% over the unconstrained ranges in the area of many data points. As expected this figure shows increasing error as K is increased. Although the least error occurs when K=1.0, this was not chosen as the best value of K,

because at high NOx levels the optimization algorithm was not able to converge on solutions. The algorithm could not converge because the parameter ranges were too restrictive to find an optimum. When K equals 3.0, the solutions are very similar to the unconstrained solutions, as expected. When a value of K equal to 2.0 is used, solutions have a slightly larger error, but the parameter ranges are very realistic and are not too conservative to defeat the purpose of the testing advisor. Using a value of K equal to 2.0 also allowed solutions to be found at most NOx levels without any problems.

Figure 45 and Figure 46 for F1 and F2 data respectively, show similar results to Figure 44. Again as K is increased the error increases. Also when values of K smaller than 2.0 are used, solutions could not be determined because the solution ranges are too restrictive. Since the error when K equals 2.0 is less than when K equals 3 and solutions could be found at most NOx levels, a value of K equal to 2.0 is chosen as the default and most appropriate constant.

Based on the results shown above, a standard deviation of two times the parameter K was found to give results which closely agreed with the manually pre-constrained results and the actual test data. For this reason, the pre-constrainer uses a constant K equal to 2.0 as its default value. This can be overridden in the *stdev.ext* file.

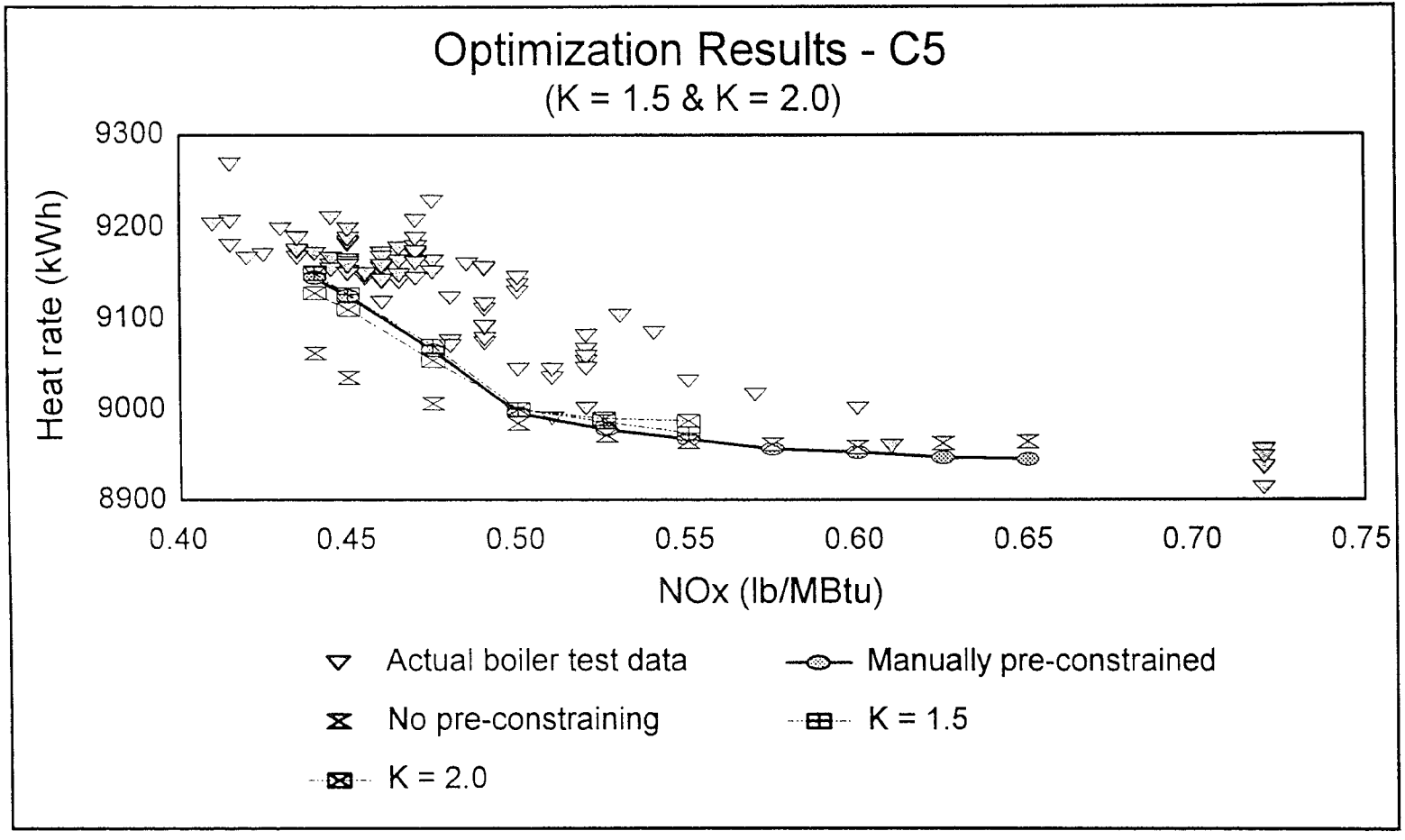

Figure 40 - C5 results (K = 1.5 & K = 2.0)

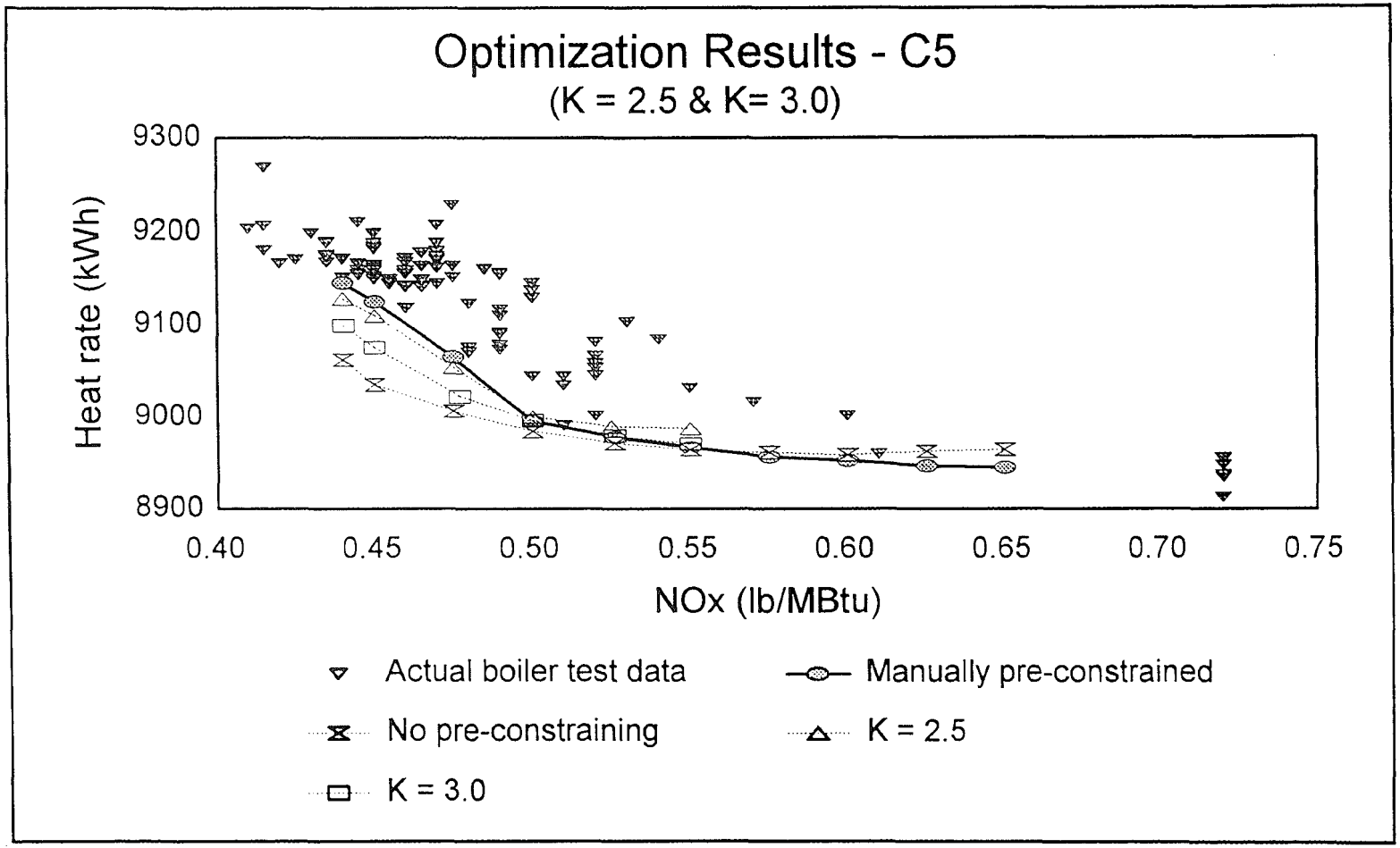

Figure 41 - C5 results (K = 2.5 & K = 3.0)

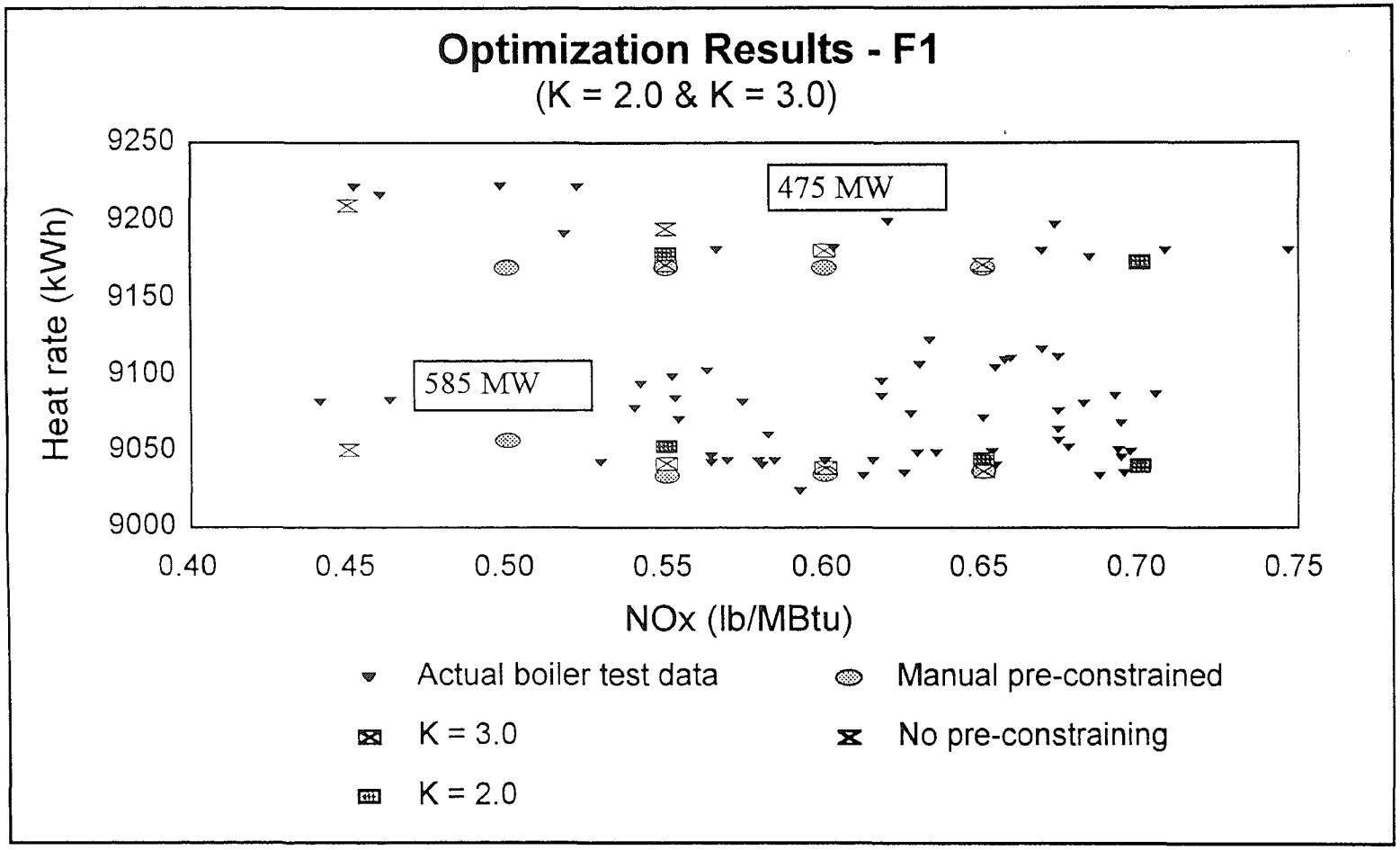

Figure 42 - F1 results (K = 2.0 & K = 3.0)

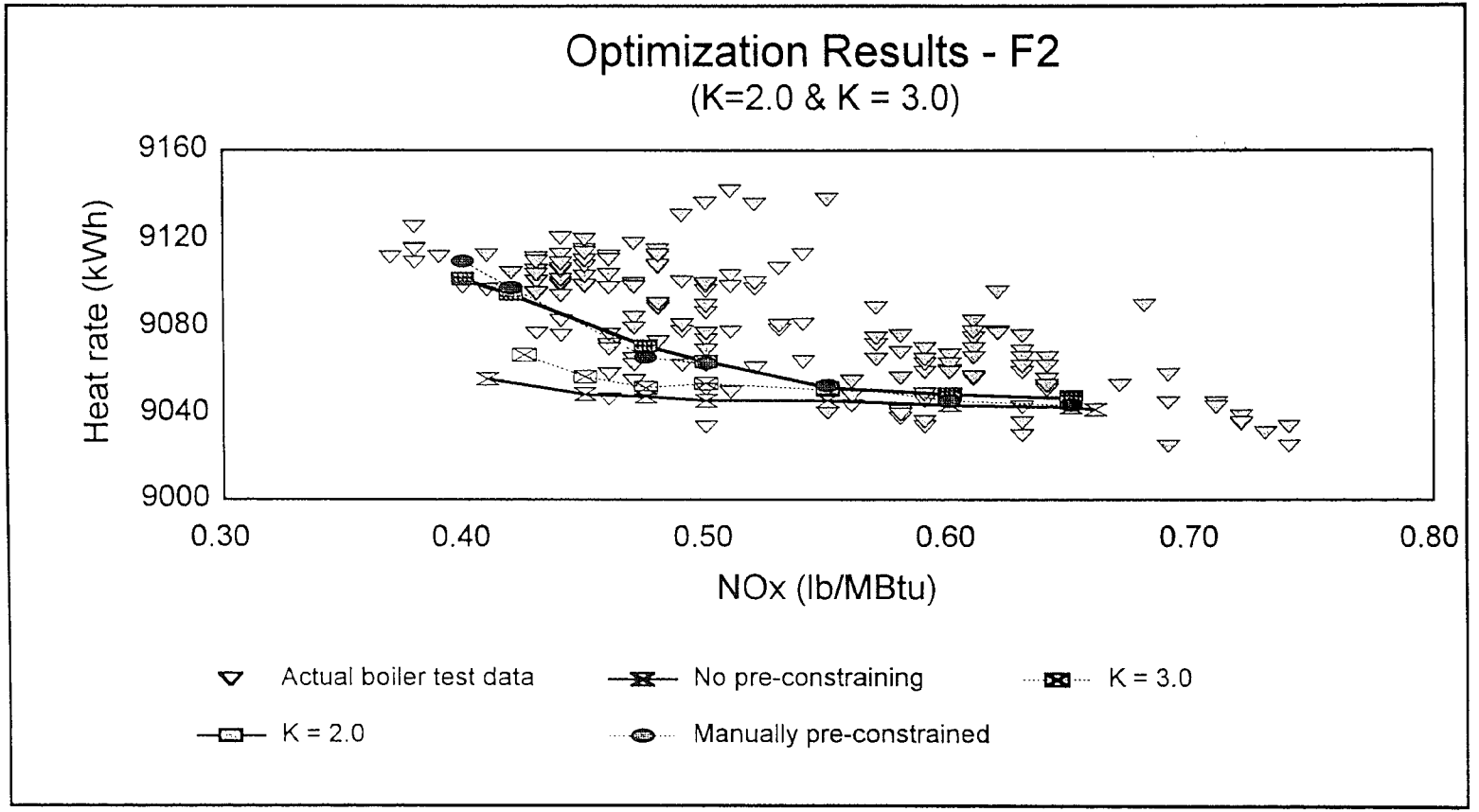

Figure 43 - F2 results (K = 2.0 & K = 3.0)

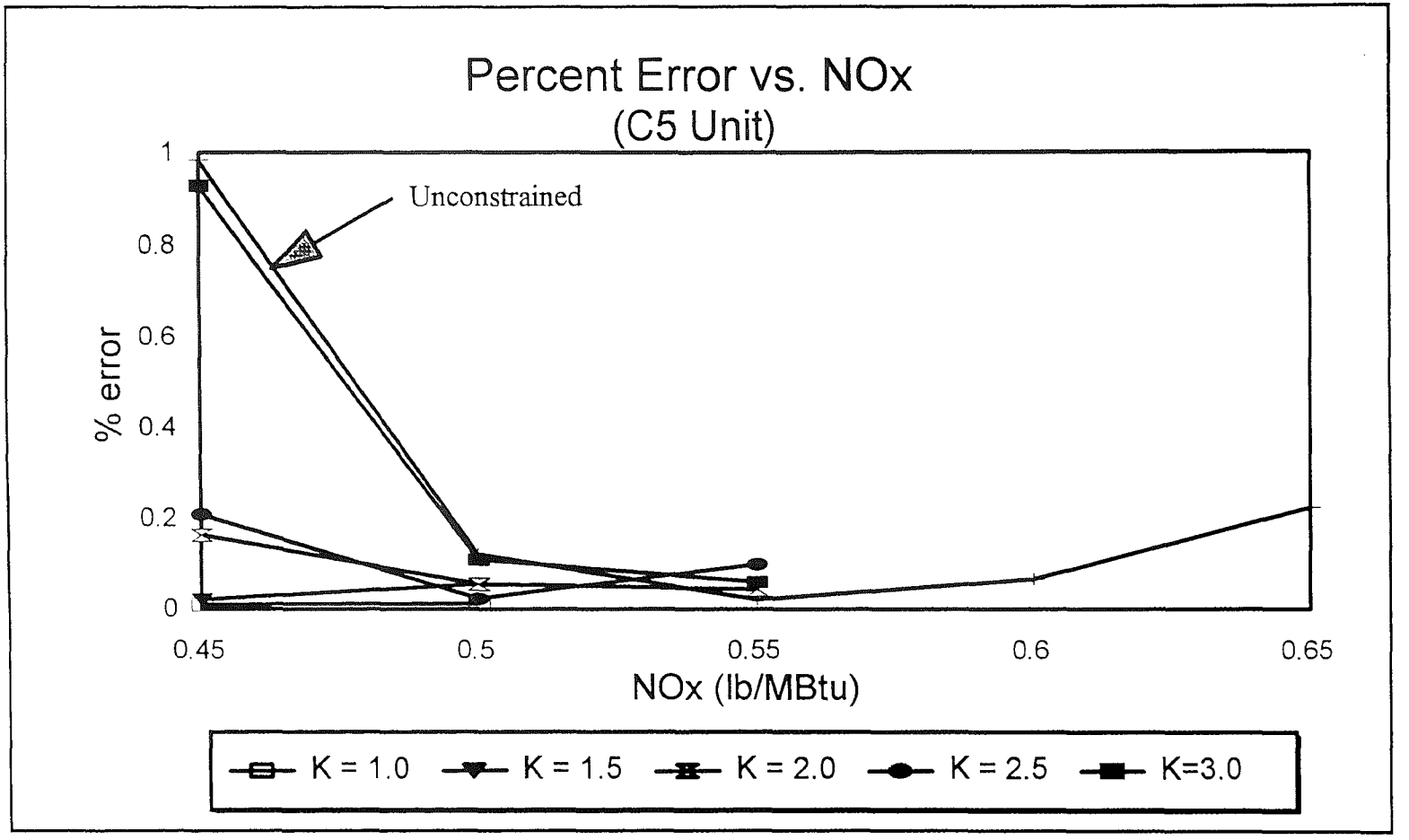

Figure 44 - Effect of K on percentage error in heat rate vs. NOx (C5 unit)

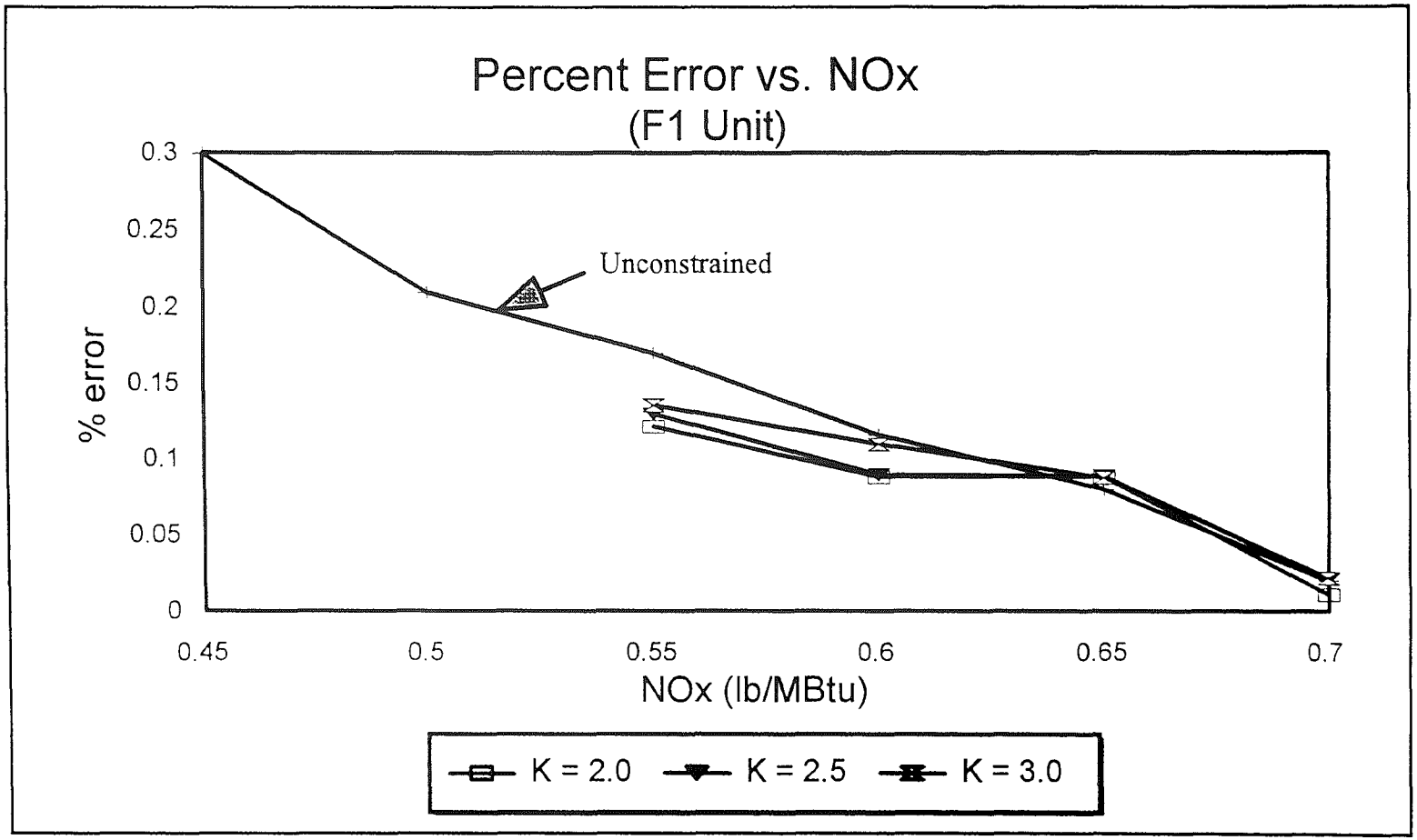

Figure 45 - Effect of K on percentage error in heat rate vs. NOx (F1 unit)

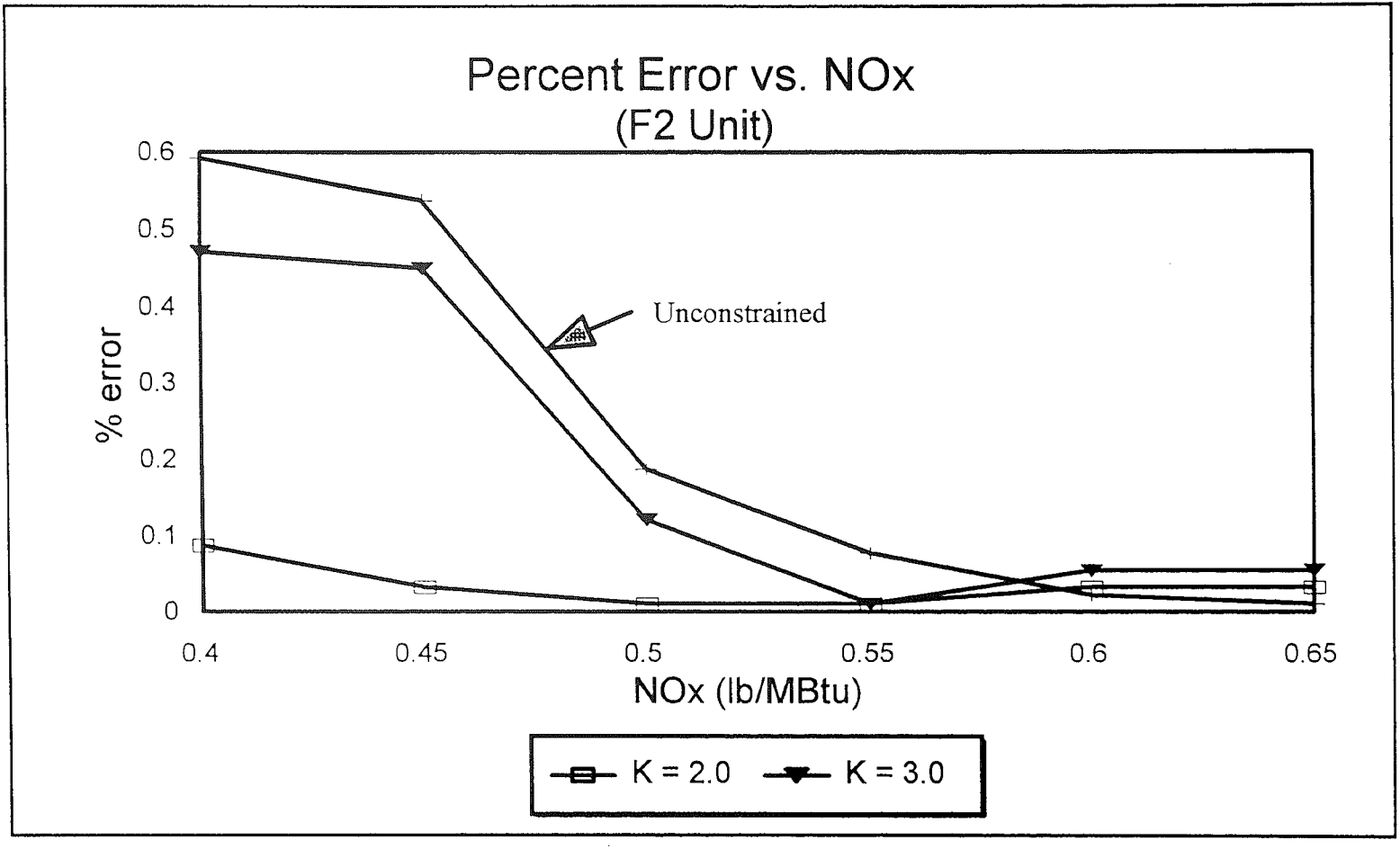

Figure 46 - Effect of K on percentage error in heat rate vs. NOx (F2 unit)

CONCLUSIONS AND RECOMMENDATIONS

Wall-Fired Testing Advisor

The previous sections discussed the design and development of the testing advisor and a detailed description of the logic used to create it. Using actual plant test data combined with generated linear relationships for each parameter as a function of NOx, the testing advisor was tested by performing four trial runs. The first trial run validated the testing advisors code and logic when the goal was to achieve an absolute minimum NOx. This trial run verified the advisor operated as intended and recommended test points which reduced NOx emissions. The second trial run validated the testing advisor code and logic when the goal was to achieve a target NOx while minimizing unit heat rate. This trial run verified the advisor operated as intended and recommended test points which reduced NOx emissions. The third trial run verified the ***significant parameter*** module operated as intended and recommended extreme tests for each parameter to determine if a significant NOx change could be achieved. The fourth trial run demonstrates the capability of handling a boiler with multiple elevations of overfire air dampers. This trial run assumed three rows of overfire air dampers existed, thus causing the overfire air damper distribution module to run. This module also ran as intended without errors and recommended tests points to reduce NOx emissions.

The wall-fired boiler testing advisor is not yet ready for use at power plants for several reasons. A data link to the plant's data acquisition system must be developed. Without this feature, it is tedious and time consuming to retrieve the data and enter it into

the advisor. Second, the present interface is the CLIPS shell which is not nearly as easy to use as a well designed Windows type user interface. Finally, the current advisor only supports full load conditions. If the advisor is to be applicable to any plant, it must accommodate low load conditions and possibly a load in between full and low.

Although, the advisor has been validated using actual test data and correlations, it should be tested on an actual boiler to determine if it is truly able to successfully test a boiler. The constraint checking has not yet been completely tested and actual values for heat rate, steam temperatures and other parameters were not tested. The advisor also should be tested on several boilers to make certain the knowledge base is complete and works on a wide range of wall-fired boilers as it was intended.

Future plans for the advisor include adding the capability to test a boiler at part load conditions. Other new modules might also be developed to test primary air flows, coal spreader position and other possible combinations which might be important. Such possible combinations include testing economizer-oxygen, shrouds and swirl settings.

Optimization Pre-Constrainer

The results shown in this thesis verify that the optimization pre-constrainer was developed and coded correctly. Throughout the testing process, the pre-constrainer operated correctly and never failed or crashed. The pre-constrainer also properly reported any errors while in operation and halted as expected.

The results indicate the pre-constrainer can accurately and reliably produce pre-constrained parameter ranges, which closely follow the actual test data. The results when

using pre-constrained ranges, are nearly the same as those found using manual pre-constraining. This verifies that the pre-constrainer is accurate and was correctly developed using the research engineers' knowledge. Finally, it was also determined that a multiplication factor, K, equal to 2.0 produced the most consistent results while not allowing unrealistic parameter ranges.

Although the pre-constrainer is complete and currently in use for conventional fired boilers, it should be tested on plant data for wall-fired boilers and implemented to work seamlessly with the wall-fired boiler testing advisor.

REFERENCES

1. The Babcock & Wilcox Company, Steam, Its Generation and Use, 40th edition. Barberton, Ohio, 1992.
2. J.G. Singer, Combustion Fossil Powered Systems. Windsor, CT: Combustion Engineering, Inc., 1981.
3. De Lorenzi, Otto, Combustion Engineering. New York, NY: Combustion Engineering, Inc., 1947.
4. Kakag, S., Boiler, Evaporators, and Condensers. New York, NY: Wiley, 1991.
5. R. Levine, D.E. Drang and B. Edelson, AI and Expert Systems, A Comprehensive Guide, 2nd edition. United States of America: McGraw-Hill Publishing Company, 1990.
6. J.P. Ignizio, Introduction to Expert Systems, The Development and Implementation of Rule-Based Expert Systems. United States of America: McGraw-Hill Publishing Company, 1990.
7. Sell, Peter S., Expert Systems - A Practical Introduction. New York, NY: Halsted Press, 1985.
8. Tzafestas, S., Expert Systems in Engineering Applications. Berlin, Germany: Springer-Verlag, 1993.
9. Puppe, F., Systematic Introduction to Expert Systems, Knowledge Representation and Problem Solving Methods. New York, NY: Springer-Verlag, 1993.

10. J. C. Giartano, CLIPS User's Guide. NASA, Lyndon B. Johnson Space Center, 1991.

11. Title 40, Code of Federal Regulations, Part 60, U.S. Government Printing Office, Washington, D. C., 1991.

12. Electric Power Research Institute, Proceedings: Expert System Applications for the Electric Power Industry. Palo Alto, CA: EPRI, June 1992.

13. Hjamarsson, A. K. and Soud, H. N., Systems for Controlling NOx from Coal Combustion. Report IEACR/30, International Energy Agency, London, 1990.

14. Lim, K.G., et al., Environment Assessment of Utility Boiler Combustion Modification NOx Control. U.S. Environmental Protection Agency, 1990.

15. Peuha, R. M., Proceedings: Environmental Effects of Nitrogen Oxides. Palo Alto, CA: EPRI, July 1989.

16. Zeldovich, J. B., Oxidation of Nitrogen in Combustion and Explosion. Academic des Sciences del URSS - Comptes Reudus, Vol. 51, No. 2.

17. Sarofim, A. F. and Posl, J. H., Kinetics of Nitrogen Oxide Formation in Premixed Laminar Flames. Presented at the Fourteenth Symposium on Combustion: Pennsylvania State University, University Park, 1972.

18. Goupy, J. L., Methods for Experimental Design. Amsterdam, The Netherlands: Elsevier Science Publishers, 1988.

19. McLean, R. A., Applied Factorial and Fractional Designs. New York, NY: Marcel Dekker, Inc., 1984.

20. Bauer, F. W. and Eberhardt, W. H., Conversion of Chalk Point Unit 2 to DB Riley Low NOx Burners. Worcester, Ma: DB Riley, Inc. 1994.

21. T. Eldredge, Summary of Chalk Point Unit 2 Test Data Analysis. Bethlehem, PA: Energy Research Center, 1994.

22. P. Lee, Application of Neural Network and Mathematical Optimization Techniques for NOx Control. Bethlehem, PA: Lehigh University, Energy Research Center, 1994.

23. D. Eskenazi, M. D'Agostini and P. Lee, Results of Extended Phase I and Phase II Low NOx Testing. Bethlehem: Energy Research Center, Report 93-500-10-16, 1993.

Vita

Steven Steele was born in Wilkes-Barre, Pennsylvania, on December 1, 1972. His parents, David and Elaine Steele, reside in Clarks Summit, Pennsylvania. He completed his high school education at Abington Heights High School in Clarks Summit and received his Bachelor's degree in mechanical engineering from Lehigh University in Bethlehem, Pennsylvania in May 1995.

END OF TITLE